SAS Publishing

SAS/ETS® 9.1 User's Guide

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS/ETS® 9.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 9.1 User's Guide

ISBN 1-59047-244-6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2004

Contents

Acknowledgments

Credits

Documentation

Editing	Virginia Clark
Technical Review	Evan L. Anderson, Brent Cohen, Allison Crutchfield, Paige Daniels, Gül Ege, Bruce Elsheimer, Donald J. Erdman, Kelly Fellingham, Phil Hanna, Kenneth N. Hightower, R. Bart Killam, Huiwen Lai, Michael J. Leonard, Mark R. Little, Kevin Meyer, Steve Morrison, Sunil Panikkath, Jim Seabolt, Michele A. Trovero, Jonathan Wang, Shirley Wang, Donna E. Woodward, Ozkan Zengin
Documentation Production	Kenneth N. Hightower, Michele A. Trovero, Tim Arnold

Software

The procedures in SAS/ETS software were implemented by members of the Analytical Solutions Division. Program development includes design, programming, debugging, support, documentation, and technical review. In the following list, the names of the developers currently supporting the procedure are listed first.

ARIMA	Rajesh Selukar, Michael J. Leonard, Terry Woodfield
AUTOREG	Jonathan Wang, John P. Sall
COMPUTAB	Michael J. Leonard, Alan R. Eaton, David F. Ross
DATASOURCE	Kelly Fellingham, Meltem Narter
ENTROPY	Donald J. Erdman
EXPAND	Michael J. Leonard, Mark R. Little
FORECAST	Michael J. Leonard, Mark R. Little, John P. Sall
LOAN	Gül Ege

MDC	Ozkan Zengin
MODEL	Donald J. Erdman, Mark R. Little, John P. Sall
PDLREG	Jonathan Wang, Leigh A. Ihnen
QLIM	Jonathan Wang
SIMLIN	Mark R. Little, John P. Sall
SPECTRA	Rajesh Selukar, Donald J. Erdman, John P. Sall
STATESPACE	Jonathan Wang, Michael J. Leonard
SYSLIN	Donald J. Erdman, Leigh A. Ihnen, John P. Sall
TIMESERIES	Michael Leonard,
TSCSREG	Jonathan Wang, Meltem Narter, Sunil Panikkath
UCM	Rajesh Selukar
VARMAX	Youngin Park
X11	Wilma S. Jackson, R. Bart Killam, Leigh A. Ihnen, Richard D. Langston
X12	Wilma S. Jackson
Time Series Forecasting System	Evan L. Anderson, Michael J. Leonard, Meltem Narter, Gül Ege
Investment Analysis System	Gül Ege, Scott Gray, Michael J. Leonard
Compiler and Symbolic Differentiation	Stacey Christian
SASEHAVR	Kelly Fellingham, Stephen Caudle
SASECRSP, SASEFAME, Data Interface Engines	Kelly Fellingham
Testing	Jackie Allen, Ming-Chun Chang, Evan Dean, Bruce Elsheimer, Tae Yoon Lee, Huiwen Lai, Audun Runde, Steve Morrison, Mark Traccarella, Xianghong Shirley Wang

Technical Support

Members	Paige Daniels, Kurt Jones, Kevin Meyer, Donna E. Woodward

Acknowledgments

Hundreds of people have helped the SAS System in many ways since its inception. The following individuals have been especially helpful in the development of the procedures in SAS/ETS software. Acknowledgments for the SAS System generally appear in Base SAS software documentation and SAS/ETS software documentation.

David Amick	Idaho Office of Highway Safety
David M. DeLong	Duke University
David Dickey	North Carolina State University
Douglas J. Drummond	Center for Survey Statistics
William Fortney	Boeing Computer Services
Wayne Fuller	Iowa State University
A. Ronald Gallant	The University North Carolina at Chapel Hill
Phil Hanser	Sacramento Municipal Utilities District
Marvin Jochimsen	Mississippi R&O Center
Ken Kraus	CRSP
George McCollister	San Diego Gas & Electric
Douglas Miller	Purdue University
Brian Monsell	U.S. Census Bureau
Robert Parks	Washington University
Gregory Sali	Idaho Office of Highway Safety
Bob Spatz	CRSP
Mary Young	Salt River Project

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

What’s New in SAS/ETS 9 and 9.1

Overview

New procedures in SAS/ETS include the following:

- The experimental ENTROPY procedure provides Generalized Maximum Entropy estimation for linear systems of equations.
- The QLIM procedure analyzes univariate and multivariate models where dependent variables take discrete values or values in a limited range.
- The TIMESERIES procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format.
- The UCM procedure provides estimation for Unobserved Component Models, also referred to as Structural Models.

Several new financial and date, time, and datetime functions have been added.

The new experimental SASEHAVR interface engine is now available to SAS/ETS for Windows users for accessing economic and financial data residing in a HAVER ANALYTICS Data Link Express (DLX) database.

New features have been added to the following SAS/ETS components:

- PROC ARIMA
- PROC EXPAND
- PROC MDC
- PROC MODEL
- PROC VARMAX
- PROC X12
- Time Series Forecasting System

Financial Functions

SAS/ETS now provides new financial functions. They are described in detail in Chapter 4, “SAS Macros and Functions.”

CUMIPMT	Returns the cumulative interest paid on a loan between the start period and the end period.
CUMPRINC	Returns the cumulative principal paid on a loan between the start period and the end period.

IPMT	Returns the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.
PMT	Returns the periodic payment for a constant payment loan or the periodic saving for a future balance.
PPMT	Returns the payment on the principal for an investment for a given period.

Date, Time, and Datetime Functions

SAS/ETS now provides the following new date, time, and datetime functions. See Chapter 3, "Date Intervals, Formats, and Functions," for more details.

INTFMT	Returns a recommended format given a date, time, or datetime interval.
INTCINDEX	Returns the cycle index given a date, time, or datetime interval and value.
INTCYCLE	Returns the date, time, or datetime interval at the next higher seasonal cycle given a date, time, or datetime interval.
INTINDEX	Returns the seasonal index given a date, time, or datetime interval and value.
INTSEA	Returns the length of the seasonal cycle given a date, time, or datetime interval.

SASEHAVR Engine

The experimental SASEHAVR interface engine gives Windows users random access to economic and financial data residing in a HAVER ANALYTICS Data Link Express (DLX) database. You can limit the range of data that is read from the time series and specify a desired conversion frequency. Start dates are recommended on the libname statement to help you save resources when processing large databases or when processing a large number of observations. You can further the subsetting of your data by using the WHERE, KEEP, or DROP statements in your DATA step. You can use the SQL procedure to create a view of your resulting SAS data set.

ARIMA Procedure

The OUTLIER statement of the ARIMA procedure has become production in SAS System 9. A new ID option that provides date labels to the discovered outliers has been added.

9.1 In the presence of embedded missing values, the new default White Noise test of residuals uses the one proposed by Stoffer and Toloi (1992), which is more appropriate.

9.1 The default forecasting algorithm when the data have embedded missing values and the model has multiple orders of differencing for the dependent series has been slightly modified. This modification usually improves the statistical properties of the forecasts.

ENTROPY Procedure

The new experimental ENTROPY procedure implements a parametric method of linear estimation based on Generalized Maximum Entropy.

Often the statistical-economic model of interest is ill-posed or underdetermined for the observed data, for example when limited data is available or acquiring data is costly. For the general linear model this can imply that high degrees of collinearity exist among explanatory variables or that there are more parameters to estimate than observations to estimate them with. These conditions lead to high variances or non-estimability for traditional GLS estimates.

The principle of maximum entropy, at the base of the ENTROPY procedure, is the foundation for an estimation methodology that is characterized by its robustness to ill-conditioned designs and its ability to fit overparameterized models.

Generalized Maximum Entropy, GME, is a means of selecting among probability distributions so as to choose the distribution that maximizes uncertainty or uniformity remaining in the distribution, subject to information already known about the distribution itself. Information takes the form of data or moment constraints in the estimation procedure. PROC ENTROPY creates a GME distribution for each parameter in the linear model, based upon support points supplied by the user. The mean of each distribution is used as the estimate of the parameter. Estimates tend to be biased, as they are a type of shrinkage estimate, but will typically portray smaller variances than OLS counterparts, making them more desirable from a mean squared error viewpoint.

PROC ENTROPY can be used to fit simultaneous systems of linear regression models, Markov models, and seemingly unrelated regression models as well as to solve pure inverse problems and unordered, multinomial choice problems. Bounds and restrictions on parameters can be specified and Wald, Likelihood ratio, and Lagrange multiplier tests can be computed. Prior information can also be supplied to enhance estimates and data. 9.1

EXPAND Procedure

The EXPAND procedure has several new transformation operators: moving product, moving rank, moving geometric mean, sequence operators, fractional differencing, Hodrick-Prescott filtering, and scaling.

The EXPAND procedure has a new option for creating time series graphics. The PLOT= option enables you to graph the input, output, and transformed time series. 9.1

MDC Procedure

The RESTRICT statement now has a new syntax and supports linear restrictions.

The new BOUNDS statement enables you to specify simple boundary constraints on the parameter estimates. You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints.

MODEL Procedure

The SMM (Simulated Method of Moments) estimation is now available as an option in the FIT statement. This method of estimation is appropriate for estimating models in which integrals appear in the objective function and these integrals can be approximated by simulation. There may be various reasons for that to happen, for example, transformation of a latent model into an observable model, missing data, random coefficients, heterogeneity, etc. A typical use of SMM is in estimating stochastic volatility models in finance, where only the stock return is observable, while the volatility process is not, and needs to be integrated out of the likelihood function. The simulation method can be used with all the estimation methods except Full Information Maximum Likelihood (FIML) in PROC MODEL. Simulated Generalized Method of Moments (SGMM) is the default estimation method.

9.1 Heteroscedastic Corrected Covariance Matrix Estimators (HCCME) have been implemented. The HCCME= option selects which correction is applied to the covariance matrix.

9.1 Instrumental variables can now be specified for specific equations rather than for all equations. This is done with expanded syntax on the INSTRUMENT statement.

QLIM Procedure

The new QLIM procedure analyzes univariate and multivariate limited dependent variable models where dependent variables take discrete values or dependent variables are observed only in a limited range of values. This procedure includes logit, probit, tobit, selection, and multivariate models. The multivariate model can contain discrete choice and limited endogenous variables as well as continuous endogenous variables.

The QLIM procedure supports the following models:

- linear regression model with heteroscedasticity
- probit with heteroscedasticity
- logit with heteroscedasticity
- tobit (censored and truncated) with heteroscedasticity
- Box-Cox regression with heteroscedasticity

- bivariate probit
- bivariate tobit
- sample selection and switching regression models
- multivariate limited dependent variables

TIMESERIES Procedure

The new TIMESERIES procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format. The procedure can perform trend and seasonal analysis on the transactions. Once the transactional data are accumulated, time domain and frequency domain analysis can be performed on the resulting time series. The procedure produces numerous graphical results related to time series analysis.

UCM Procedure

The new UCM procedure, experimental in SAS System 9, is production in SAS 9.1. **9.1**
You can use this procedure to analyze and forecast equally spaced univariate time series data using Unobserved Components Models (UCM).

The UCMs can be regarded as regression models where, apart from the usual regression variables, the model consists of components such as trend, seasonals, and cycles. In time series literature UCMs are also referred to as Structural Models. The different components in a UCM can be modeled separately and are customized to represent salient features of a given time series. The analysis provides separate in-sample and out of sample estimates (forecasts) of these component series. In particular, model-based seasonal decomposition and seasonal adjustment of the dependent series is easily available. The distribution of errors in the model is assumed to be Gaussian and the model parameters are estimated by maximizing the Gaussian likelihood. The UCM procedure can handle missing values in the dependent series.

The domains of applicability of PROC UCM and PROC ARIMA are virtually identical; however, decomposition of a series in features such as trend, seasonals, and cycles is more convenient in PROC UCM. A seasonal decomposition of a time series can also be obtained using other procedures, for example, PROC X12. However, these seasonal decompositions generally do not take into account regression and other effects and are not model based. The seasonal decomposition in PROC UCM is based on a comprehensive model, providing all the advantages of model diagnostics.

VARMAX Procedure

The VARMAX procedure now provides the following features:

- The ECTREND option is available in the ECM=( ) option of the MODEL statement to fit the VECM(p) with a restriction on the drift. The ECTREND option is ignored when either the NSEASON or NOINT option is specified.
- You can now use the DFTEST option at multiple lags. For example, DFTEST=(DLAG=(1)(12)) provides the Dickey-Fuller regular unit root test and seasonal unit root test. If the TREND= option is specified, the seasonal unit root test is not available.
- The DYNAMIC option is added to the PRINT=( ) option. This representation displays the contemporaneous relationships among the components of the vector time series.
- The CORRX, CORRY, COVPE, COVX, COVY, DECOMPOSE, IARR, IMPULSE, IMPULSX, PARCOEF, PCANCORR, and PCORR options can be used with the number in parentheses in the PRINT=( ) option. For example, you can use CORRX or CORRX(*number*). The options print the number of lags specified by *number*. The default is the number of lags specified by the LAGMAX=*number*.
- The subset BVAR model is now available.
- The statistics for the one lagged coefficient matrix are removed in the ECM.
- The last columns of the BETA and ALPHA are removed in the COINTTEST option when the NOINT option is not specified.
- The long variable names are available in the model parameter estimation table.
- The schematic representation of the estimates that shows the significance of the parameters is now available.
- Two new ODS Tables, ParameterGraph and GARCHParameterGraph, are added.

9.1 Many ODS table names have been changed.

X12 Procedure

The X12 procedure default behavior has changed with regard to missing leading and trailing values. Previously the default was not to trim leading/trailing missing values from the series. This made it difficult to process multiple series within a data set when the series had differing spans. Now the default is to trim leading and trailing missing values. The new NOTRIMMISS option provides the old default behavior; when NOTRIMMISS is specified, PROC X12 will automatically generate missing value regressors for any missing value within the span of the series, including leading and trailing missing values.

The following statements and options are new:

- The AUTOMDL statement uses the TRAMO method based on the work of Gomez and Maravall (1997a and 1997b) to automatically select the ARIMA part of a regARIMA model for the time series.
- The OUTLIER statement automatically detects additive, level shift, and temporary change outliers in the time series. After the outliers are identified, the appropriate regression variables are incorporated into the model.
- The MAXITER and TOL options of the ESTIMATE statement provide additional control over the convergence of the nonlinear estimation.
- The ITPRINT and PRINTERR options of the ESTIMATE statement enable you to examine the iterations history of the nonlinear estimation.
- The FINAL and FORCE options of the X11 statement enable you to control the final seasonally adjusted series. The FINAL option specifies whether outlier, level shift, and temporary change effects should be removed from the final seasonally adjusted series. The FORCE option specifies whether or not the yearly totals of final seasonally adjusted series match the totals of the original series. **9.1**

Time Series Forecasting System

Enhancements to this graphical point-and-click system provide new kinds of forecasting models, better ways to customize lists of models, greater flexibility in sharing projects over a network, and support for graphical and tabular Web reports:

- The Factored ARIMA Model Specification window provides a general purpose interface for specifying ARIMA models. You can specify any number of factors and select the AR and MA lags to include in each factor. This makes it easy to model series with unusual and/or multiple seasonal cycles.
- Improvements to the Model Selection List Editor window enable you to open alternate model lists included with the software as well as user defined model lists. You can create a new model list, open an existing model list, modify it, use it to replace the current list, append it to the current list, save it in a catalog, assign it to a project, or assign it as a user default list for newly created projects. **9.1**

Several new ARIMA and dynamic regression model lists are provided. You can combine these into large sets for automatic model selection and select from them to create the best set of candidate models for your data.

9.1

- Project options are no longer stored exclusively in the SASUSER library. You can use any path to which you have write access by assigning the libname TSFSUSER. The system prompts you for this path if you do not have write access to the SASUSER library.

9.1

- The Series Viewer and Model Viewer support saving graphs and tables via the Output Delivery System (ODS). Select the "Use Output Delivery System" option in the Save As dialog to create html pages and corresponding gif files. You can access and organize these using the ODS Results window, display them automatically in your browser (depending on your results preferences settings), or publish them via the Internet or an intranet. You can also create other forms of output by providing your own ODS statements.
- The Time Series Viewer and Time Series Forecasting System can now be started from a macro submitted from the Program Editor or the Enhanced Editor. The FORECAST and TSVIEW macros accept the same arguments as the FORECAST and TSVIEW commands. You can use the FORECAST macro to generate and submit any number of independent unattended forecasting runs from a data step program.

References

Gomez, V. and A. Maravall (1997a), "Program TRAMO and SEATS: Instructions for the User, Beta Version," Banco de Espana.

Gomez, V. and A. Maravall (1997b), "Guide for Using the Programs TRAMO and SEATS, Beta Version," Banco de Espana.

Stoffer, D. and Toloi, C. (1992), "A Note on the Ljung-Box-Pierce Portmanteau Statistic with Missing Data," *Statistics & Probability Letters* 13, 391-396.

Part 1

General Information

Contents

Chapter 1
Introduction

Chapter Contents

Chapter 1
Introduction

Overview of SAS/ETS Software

SAS/ETS software, a component of the SAS System, provides SAS procedures for

- econometric analysis
- time series analysis
- time series forecasting
- systems modeling and simulation
- discrete choice analysis
- analysis of qualitative and limited dependent variable models
- seasonal adjustment of time series data
- financial analysis and reporting
- access to economic and financial databases
- time series data management

In addition to SAS procedures, SAS/ETS software also includes interactive environments for time series forecasting and investment analysis.

Uses of SAS/ETS Software

SAS/ETS software provides tools for a wide variety of applications in business, government, and academia. Major uses of SAS/ETS procedures are economic analysis, forecasting, economic and financial modeling, time series analysis, financial reporting, and manipulation of time series data.

The common theme relating the many applications of the software is time series data: SAS/ETS software is useful whenever it is necessary to analyze or predict processes that take place over time or to analyze models that involve simultaneous relationships.

Although SAS/ETS software is most closely associated with business and economics, time series data also arise in many other fields. SAS/ETS software is useful whenever time dependencies, simultaneous relationships, or dynamic processes complicate data analysis. For example, an environmental quality study might use SAS/ETS software's time series analysis tools to analyze pollution emissions data. A pharmacokinetic study might use SAS/ETS software's features for nonlinear systems to model the dynamics of drug metabolism in different tissues.

The diversity of problems for which econometrics and time series analysis tools are needed is reflected in the applications reported by SAS users. The following listed items are some applications of SAS/ETS software presented by SAS users at past annual conferences of the SAS Users Group International (SUGI).

- forecasting college enrollment (Calise and Earley 1997)
- fit a pharmacokinetic model (Morelock et al. 1995)
- testing interaction effect in reducing SIDS (Fleming, Gibson, and Fleming 1996)
- forecasting operational indices to measure productivity changes (McCarty 1994)
- spectral decomposition and reconstruction of nuclear plant signals (Hoyer and Gross 1993)
- estimating parameters for the CES-Translog model (Hisnanick 1993)
- applying econometric analysis for mass appraisal of real property (Amal and Weselowski 1993)
- forecasting telephone usage data (Fishetti, Heathcote, and Perry 1993)
- forecasting demand and utilization of inpatient hospital services (Hisnanick 1992)
- using conditional demand estimation to determine electricity demand (Keshani and Taylor 1992)
- estimating tree biomass for measurement of forestry yields (Parresol and Thomas 1991)
- evaluating the theory of input separability in the production function of U.S. manufacturing (Hisnanick 1991)
- forecasting dairy milk yields and composition (Benseman 1990)
- predicting the gloss of coated aluminum products subject to weathering (Khan 1990)
- learning curve analysis for predicting manufacturing costs of aircraft (Le Bouton 1989)
- analyzing Dow Jones stock index trends (Early, Sweeney, and Zekavat 1989)
- analyzing the usefulness of the composite index of leading economic indicators for forecasting the economy (Lin and Myers 1988)

Contents of SAS/ETS Software

Procedures

SAS/ETS software includes the following SAS procedures:

ARIMA	ARIMA (Box-Jenkins) and ARIMAX (Box-Tiao) modeling and forecasting
AUTOREG	regression analysis with autocorrelated or heteroscedastic errors and ARCH and GARCH modeling
COMPUTAB	spreadsheet calculations and financial report generation
DATASOURCE	access to financial and economic databases

ENTROPY	maximum entropy-based regression
EXPAND	time series interpolation and frequency conversion, and transformation of time series
FORECAST	automatic forecasting
LOAN	loan analysis and comparison
MDC	multinomial discrete choice analysis
MODEL	nonlinear simultaneous equations regression and nonlinear systems modeling and simulation
PDLREG	polynomial distributed lag regression (Almon lags)
QLIM	qualitative and limited dependent variable analysis
SIMLIN	linear systems simulation
SPECTRA	spectral and cross spectral analysis
STATESPACE	state space modeling and automated forecasting of multivariate time series
SYSLIN	linear simultaneous equations models
TIMESERIES	analysis of time-stamped transactional data
TSCSREG	time series cross-sectional regression analysis
UCM	unobserved components analysis of time series
VARMAX	vector autoregressive and moving-average modeling and forecasting
X11	seasonal adjustment (Census X-11 and X-11 ARIMA)
X12	seasonal adjustment (Census X-12 ARIMA)

Macros

SAS/ETS software includes the following SAS macros:

%AR	generates statements to define autoregressive error models for the MODEL procedure
%BOXCOXAR	investigates Box-Cox transformations useful for modeling and forecasting a time series
%DFPVALUE	computes probabilities for Dickey-Fuller test statistics
%DFTEST	performs Dickey-Fuller tests for unit roots in a time series process
%LOGTEST	tests to see if a log transformation is appropriate for modeling and forecasting a time series
%MA	generates statements to define moving average error models for the MODEL procedure
%PDL	generates statements to define polynomial distributed lag models for the MODEL procedure

These macros are part of the SAS AUTOCALL facility and are automatically available for use in your SAS program. Refer to *SAS Macro Language: Reference* for information about the SAS macro facility.

The Time Series Forecasting System

In addition to SAS procedures and macros, SAS/ETS software also includes an interactive forecasting user interface. This user interface was developed with SAS/AF software and uses PROC ARIMA to perform time series forecasting. The TSF system makes it easy to forecast time series and provides many features for graphical data exploration and graphical comparisons of forecasting models and forecasts. (SAS/GRAPH is required to use the graphical features of the system.)

The Investment Analysis System

The Investment Analysis system is an interactive environment for the time-value of money of a variety of investments. Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

Some of the features of SAS/ETS software are also available through menu driven interfaces provided by SAS/ASSIST software. (Both SAS/ASSIST software and SAS/ETS software must be licensed for you to use these features.)

The following components of SAS/ASSIST software enable you to use SAS/ETS procedures through a menu interface:

- loan analysis (uses PROC LOAN)
- regression with correction for autocorrelation (uses PROC AUTOREG)
- seasonal adjustment (uses PROC X11)
- convert frequency of time series data (uses PROC EXPAND)

About This Book

This book is a user's guide to SAS/ETS software. Since SAS/ETS software is a part of the SAS System, this book assumes that you are familiar with Base SAS software and have the books *SAS Language: Reference* and *SAS Procedures Guide* available for reference. It also assumes that you are familiar with SAS data sets, the SAS DATA step, and with basic SAS procedures such as PROC PRINT and PROC SORT. Chapter 2, "Working with Time Series Data," in this book summarizes the aspects of Base SAS software most relevant to the use of SAS/ETS software.

Chapter Organization

Following a brief What's New section, this book is divided into four major parts. "Part One" contains general information to aid you in working with SAS/ETS Software. "Part Two" is the "Procedure Reference" that is comprised of chapters that explain the SAS procedures that make up SAS/ETS software. "Part Three" is the reference for the Time Series Forecasting System, an interactive forecasting menu system that uses PROC ARIMA to perform time series forecasting. Finally, "Part Four" is the reference for the Investment Analysis System.

The new features added to SAS/ETS software since the publication of *SAS/ETS Software: Changes and Enhancements for Release 8.2* are summarized in “What’s New in SAS/ETS 9 and 9.1.” If you have used SAS/ETS software in the past, you may want to skim this chapter to see what’s new.

“Part One” contains the following chapters.

Chapter 1, the current chapter, provides an overview of SAS/ETS software and summarizes related SAS Institute publications, products, and services.

Chapter 2, “Working with Time Series Data,” discusses the use of SAS data management and programming features for time series data.

Chapter 3, “Date Intervals, Formats, and Functions,” summarizes the time intervals, date and datetime informats, date and datetime formats, and date and datetime functions available in the SAS System.

Chapter 4, “SAS Macros and Functions,” documents SAS macros and DATA step financial functions provided with SAS/ETS software. The macros use SAS/ETS procedures to perform Dickey-Fuller tests, test for the need for log transformations, or select optimal Box-Cox transformation parameters for time series data.

Chapter 5, “The SASECRSP Interface Engine,” documents the SASECRSP interface engine that enables SAS users to access and process time series data residing in CRSPAccess data files, and provides a seamless interface between CRSP and SAS data processing.

Chapter 6, “The SASEFAME Interface Engine,” documents the SASEFAME interface engine that enables SAS users to access and process time series data residing in a FAME database, and provides a seamless interface between FAME and SAS data processing.

Chapter 7, “The SASEHAVR Interface Engine,” documents the SASEHAVR interface engine that provides Windows users random access to economic and financial data residing in a HAVER ANALYTICS Data Link Express (DLX) database.

Chapter 8, “Using the Output Delivery System,” provides an introduction to the Output Delivery System (ODS).

Chapter 9, “Statistical Graphics Using ODS,” provides an introduction to the experimental graphics extension to the Output Delivery System.

Chapter 10, “Nonlinear Optimization Methods,” documents the NonLinear Optimization subsystem used by some ETS procedures to perform nonlinear optimization tasks.

“Part Two” contains the chapters that explain the SAS procedures that make up SAS/ETS software. These chapters appear in alphabetical order by procedure name.

The chapters documenting the SAS/ETS procedures are organized as follows:

1. Each chapter begins with an “Overview” section that gives a brief description of the procedure.

2. The “Getting Started” section provides a tutorial introduction on how to use the procedure.
3. The “Syntax” section is a reference to the SAS statements and options that control the procedure.
4. The “Details” section discusses various technical details.
5. The “Examples” section contains examples of the use of the procedure.
6. The “References” section contains technical references on methodology.

“Part Three” contains the chapters that document the features of the Time Series Forecasting System, while the features of the Investment Analysis System are documented in “Part Four.”

Typographical Conventions

This book uses several type styles for presenting information. The following list explains the meaning of the typographical conventions used in this book:

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.
UPPERCASE BOLD	is used in the “Syntax” sections’ initial lists of SAS statements and options.
oblique	is used for user-supplied values for options in the syntax definitions. In the text, these values are written in *italic*.
helvetica	is used for the names of variables and data sets when they appear in the text.
bold	is used to refer to matrices and vectors, and to refer to commands (e.g **end** or **cd**.)
italic	is used for terms that are defined in the text, for emphasis, and for references to publications.
`monospace`	is used for example code. In most cases, this book uses lowercase type for SAS code.

Options Used in Examples

Output of Examples

For each example, the procedure output is numbered consecutively starting with 1, and each output is given a title. Each page of output produced by a procedure is enclosed in a box.

Most of the output shown in this book is produced with the following SAS System options:

```
options linesize=80 pagesize=200 nonumber nodate;
```

The template STATDOC.TPL is used to create the HTML output that appears in the online (CD) version. A style template controls stylistic HTML elements such as colors, fonts, and presentation attributes. The style template is specified in the ODS HTML statement as follows:

```
ODS HTML style=statdoc;
```

If you run the examples, you may get slightly different output. This is a function of the SAS System options used and the precision used by your computer for floating-point calculations.

Graphics Options

The examples that contain graphical output are created with a specific set of options and symbol statements. The code you see in the examples creates the color graphics that appear in the online (CD) version of this book. A slightly different set of options and statements is used to create the black-and-white graphics that appear in the printed version of the book.

If you run the examples, you may get slightly different results. This may occur because not all graphic options for color devices translate directly to black-and-white output formats. For complete information on SAS/GRAPH software and graphics options, refer to *SAS/GRAPH Software: Reference*.

The following GOPTIONS statement is used to create the online (color) version of the graphic output.

```
filename GSASFILE  '<file-specification>';

goptions reset=all
         gaccess=GSASFILE    gsfmode=replace
         fileonly
         transparency        dev = gif
         ftext = swiss       lfactor = 1
         htext = 4.0pct      htitle = 4.5pct
         hsize = 5.5in       vsize = 3.5in
         noborder            cback = white
         horigin = 0in       vorigin = 0in ;
```

The following GOPTIONS statement is used to create the black-and-white version of the graphic output, which appears in the printed version of the manual.

```
filename GSASFILE  '<file-specification>';

goptions reset=all
         gaccess=GSASFILE    gsfmode=replace
         fileonly
```

```
dev = pslepsf
ftext = swiss        lfactor = 1
htext = 3.0pct       htitle = 3.5pct
hsize = 5.5in        vsize = 3.5in
border               cback = white
horigin = 0in        vorigin = 0in;
```

In most of the online examples, the plot symbols are specified as follows:

```
symbol1 value=dot color=white height=3.5pct;
```

The SYMBOL*n* statements used in online examples order the symbol colors as follows: white, yellow, cyan, green, orange, blue, and black.

In the examples appearing in the printed manual, symbol statements specify COLOR=BLACK and order the plot symbols as follows: dot, square, triangle, circle, plus, x, diamond, and star.

Where to Turn for More Information

This section describes other sources of information about SAS/ETS software.

Accessing the SAS/ETS Sample Library

The SAS/ETS Sample Library includes many examples that illustrate the use of SAS/ETS software, including the examples used in this documentation. To access these sample programs, select **Help** from the menu and select **SAS Help and Documentation**. From the Contents list, choose **Learning to Use SAS** and then **Sample SAS Programs**.

Online Help System

You can access online help information about SAS/ETS software in two ways, depending on whether you are using the SAS windowing environment in the command line mode or the pull-down menu mode.

If you are using a command line, you can access the SAS/ETS help menus by typing **help** on the SAS windowing environment command line. Or you can or issue the command **help ARIMA** (or another procedure name) to bring up the help for that particular procedure.

If you are using the SAS windowing environment pull-down menus, you can pull-down the **Help** menu and make the following selections:

- SAS Help and Documentation
- Learning to Use SAS in the Contents list
- SAS Products
- SAS/ETS

The content of the Online Help System follows closely the one of this book.

Other Related SAS Institute Publications

In addition to this user's guide, SAS Institute publishes other books on using SAS/ETS software. The following books are companions to this user's guide:

- *SAS/ETS Software: Applications Guide 1, Version 6, First Edition*
- *SAS/ETS Software: Applications Guide 2, Version 6, First Edition*

The first volume, *SAS/ETS Software: Applications Guide 1*, discusses features of SAS/ETS software for time series modeling and forecasting, financial reporting, and loan analysis. The second volume, *SAS/ETS Software: Applications Guide 2*, discusses features of SAS/ETS software for econometric modeling and simulation.

Forecasting Examples for Business and Economics Using the SAS System, discusses forecasting using SAS/ETS software.

SAS Institute Short Courses

SAS Institute offers the following short course on using SAS/ETS software:

Introduction to Time Series Forecasting Using SAS/ETS Software is a Level III course is designed for statisticians, economists, business planners, inventory managers, market researchers, and others who analyze time series data and need to forecast time series data. This course uses the Time Series Forecasting System (TSFS) and the SAS/ETS procedures ARIMA, FORECAST, and EXPAND. After completing this course, you should be able to

- preprocess time series data using SAS date, time, and mathematical DATA step functions
- impute missing or invalid values in time series data using a variety of methods
- recognize and understand the basic components of time series data, including trend and seasonality
- forecast individual time series using regression, exponential smoothing, ARIMA, and composite models
- implement procedures for the automatic generation of forecasts for a large number of time series
- produce forecasts using data collected at different summary levels (for example, SKU, product group, product line, business line)
- produce effective graphics presentations of forecasts
- evaluate the accuracy of forecast models
- select from a variety of competing models

SAS Institute Technical Support Services

As with all SAS Institute products, the SAS Institute Technical Support staff is available to respond to problems and answer technical questions regarding the use of SAS/ETS software.

Major Features of SAS/ETS Software

The following sections briefly summarize major features of SAS/ETS software. See the chapters on individual procedures for more detailed information.

Discrete Choice and Qualitative and Limited Dependent Variable Analysis

The MDC procedure provides maximum likelihood (ML) or simulated maximum likelihood estimates of multinomial discrete choice models in which the choice set consists of unordered multiple alternatives. The MDC procedure supports the following models and features:

- conditional logit
- nested logit
- heteroscedastic extreme value
- multinomial probit
- mixed logit
- pseudo-random or quasi-random numbers for simulated maximum likelihood estimation
- bounds imposed on the parameter estimates
- linear restrictions imposed on the parameter estimates
- SAS data set containing predicted probabilities and linear predictor ($\mathbf{x}'\boldsymbol{\beta}$) values
- goodness-of-fit measures including
  - likelihood ratio
  - Aldrich-Nelson
  - Cragg-Uhler 1
  - Cragg-Uhler 2
  - Estrella
  - Adjusted Estrella
  - McFadden’s LRI
  - Veall-Zimmermann
  - Akaike Information Criterion (AIC)
  - Schwarz Criterion

The QLIM procedure analyzes univariate and multivariate limited dependent variable models where dependent variables take discrete values or dependent variables are observed only in a limited range of values. This procedure includes logit, probit, tobit, and general simultaneous equations models. The QLIM procedure supports the following models:

- linear regression model with heteroscedasticity
- probit with heteroscedasticity
- logit with heteroscedasticity
- tobit (censored and truncated) with heteroscedasticity
- Box-Cox regression with heteroscedasticity
- bivariate probit
- bivariate tobit

Regression with Autocorrelated and Heteroscedastic Errors

The AUTOREG procedure provides regression analysis and forecasting of linear models with autocorrelated or heteroscedastic errors. The AUTOREG procedure includes the following features:

- estimation and prediction of linear regression models with autoregressive errors
- any order autoregressive or subset autoregressive process
- optional stepwise selection of autoregressive parameters
- choice of the following estimation methods:
  - exact maximum likelihood
  - exact nonlinear least squares
  - Yule-Walker
  - iterated Yule-Walker
- tests for any linear hypothesis involving the structural coefficients
- restrictions for any linear combination of the structural coefficients
- forecasts with confidence limits
- estimation and forecasting of ARCH (autoregressive conditional heteroscedasticity), GARCH (generalized autoregressive conditional heteroscedasticity), I-GARCH (integrated GARCH), E-GARCH (exponential GARCH), and GARCH-M (GARCH in mean) models
- ARCH and GARCH models can be combined with autoregressive models, with or without regressors
- estimation and testing of general heteroscedasticity models
- variety of model diagnostic information including
  - autocorrelation plots

  - partial autocorrelation plots
  - Durbin-Watson test statistic and generalized Durbin-Watson tests to any order
  - Durbin *h* and Durbin *t* statistics
  - Akaike information criterion
  - Schwarz information criterion
  - tests for ARCH errors
  - Ramsey's RESET test
  - Chow and PChow tests
  - Phillips-Perron stationarity test
  - CUSUM and CUMSUMSQ statistics

- exact significance levels (*p*-values) for the Durbin-Watson statistic
- embedded missing values

Simultaneous Systems Linear Regression

The SYSLIN and ENTROPY procedures provide regression analysis of a simultaneous system of linear equations. The SYSLIN procedure includes the following features:

- estimation of parameters in simultaneous systems of linear equations
- full range of estimation methods including
  - ordinary least squares (OLS)
  - two-stage least squares (2SLS)
  - three-stage least squares (3SLS)
  - iterated 3SLS
  - seemingly unrelated regression (SUR)
  - iterated SUR
  - limited-information maximum-likelihood (LIML)
  - full-information maximum-likelihood (FIML)
  - minimum-expected-loss (MELO)
  - general K-class estimators
- weighted regression
- any number of restrictions for any linear combination of coefficients, within a single model or across equations
- tests for any linear hypothesis, for the parameters of a single model or across equations
- wide range of model diagnostics and statistics including
  - usual ANOVA tables and R^2 statistics
  - Durbin-Watson statistics

  - standardized coefficients
  - test for over-identifying restrictions
  - residual plots
  - standard errors and T tests
  - covariance and correlation matrices of parameter estimates and equation errors

- predicted values, residuals, parameter estimates, and variance-covariance matrices saved in output SAS data sets

The ENTROPY procedure includes the following features:

- generalized maximum entropy (GME) estimation
- generalized cross entropy (GCE) estimation
- maximum entropy SUR (MESUR) estimation
- pure inverse estimation
- estimation of parameters in simultaneous systems of linear equations
- weighted regression
- any number of restrictions for any linear combination of coefficients, within a single model or across equations
- tests for any linear hypothesis, for the parameters of a single model or across equations

Linear Systems Simulation

The SIMLIN procedure performs simulation and multiplier analysis for simultaneous systems of linear regression models. The SIMLIN procedure includes the following features:

- reduced form coefficients
- interim multipliers
- total multipliers
- dynamic forecasts and simulations
- goodness-of-fit statistics
- processes equation system coefficients estimated by the SYSLIN procedure

Polynomial Distributed Lag Regression

The PDLREG procedure provides regression analysis for linear models with polynomial distributed (Almon) lags. The PDLREG procedure includes the following features:

- any number of regressors may enter as a polynomial lag distribution, and any number of covariates may be used
- any order lag length and degree polynomial for lag distribution may be used
- optional upper and lower endpoint restrictions
- any number of linear restrictions may be placed on covariates
- option to repeat analysis over a range of degrees for the lag distribution polynomials
- support for autoregressive errors to any lag
- forecasts with confidence limits

Nonlinear Systems Regression and Simulation

The MODEL procedure provides parameter estimation, simulation, and forecasting of dynamic nonlinear simultaneous equation models. The MODEL procedure includes the following features:

- nonlinear regression analysis for systems of simultaneous equations, including weighted nonlinear regression
- full range of parameter estimation methods including
  - nonlinear ordinary least squares (OLS)
  - nonlinear seemingly unrelated regression (SUR)
  - nonlinear two-stage least squares (2SLS)
  - nonlinear three-stage least squares (3SLS)
  - iterated SUR
  - iterated 3SLS
  - generalized method of moments (GMM)
  - nonlinear full information maximum likelihood (FIML)
  - simulated method of moments (SMM)
- supports dynamic multi-equation nonlinear models of any size or complexity
- uses the full power of the SAS programming language for model definition, including left-hand side expressions
- hypothesis tests of nonlinear functions of the parameter estimates
- linear and nonlinear restrictions of the parameter estimates
- bounds imposed on the parameter estimates
- computation of estimates and standard errors of nonlinear functions of the parameter estimates
- estimation and simulation of Ordinary Differential Equations (ODE's)
- vector autoregressive error processes and polynomial lag distributions easily specified for the nonlinear equations
- variance modeling (ARCH, GARCH, and others)

- computes goal-seeking solutions of nonlinear systems to find input values needed to produce target outputs
- dynamic, static, or *n*-period-ahead-forecast simulation modes
- simultaneous solution or single equation solution modes
- Monte Carlo simulation using parameter estimate covariance and across-equation residuals covariance matrices or user specified random functions
- a variety of diagnostic statistics including
  - model R^2 statistics
  - general Durbin-Watson statistics and exact p-values
  - asymptotic standard errors and T tests
  - first stage R^2 statistics
  - covariance estimates
  - collinearity diagnostics
  - simulation goodness-of-fit statistics
  - Theil inequality coefficient decompositions
  - Theil relative change forecast error measures
  - heteroscedasticity tests
  - Godfrey test for serial correlation
  - Chow tests
- block structure and dependency structure analysis for the nonlinear system
- listing and cross reference of fitted model
- automatic calculation of needed derivatives using exact analytic formula
- efficient sparse matrix methods used for model solution; choice of other solution methods
- model definition, parameter estimation, simulation, and forecasting may be performed interactively in a single SAS session or models can also be stored in files and reused and combined in later runs

ARIMA (Box-Jenkins) and ARIMAX (Box-Tiao) Modeling and Forecasting

The ARIMA procedure provides the identification, parameter estimation, and forecasting of autoregressive integrated moving average (Box-Jenkins) models, seasonal ARIMA models, transfer function models, and intervention models. The ARIMA procedure includes the following features:

- complete ARIMA (Box-Jenkins) modeling with no limits on the order of autoregressive or moving average processes
- model identification diagnostics, include the following:
  - autocorrelation function
  - partial autocorrelation function

  - inverse autocorrelation function
  - cross-correlation function
  - extended sample autocorrelation function
  - minimum information criterion for model identification
  - squared canonical correlations

- stationarity tests
- outlier detection
- intervention analysis
- regression with ARMA errors
- transfer function modeling with fully general rational transfer functions
- seasonal ARIMA models
- ARIMA model-based interpolation of missing values
- several parameter estimation methods including
  - exact maximum likelihood
  - conditional least squares
  - exact nonlinear unconditional least squares
- forecasts and confidence limits for all models
- forecasting tied to parameter estimation methods: finite memory forecasts for models estimated by maximum likelihood or exact nonlinear least squares methods and infinite memory forecasts for models estimated by conditional least squares
- a variety of model diagnostic statistics including
  - Akaike's information criterion (AIC)
  - Schwarz's Bayesian criterion (SBC or BIC)
  - Box-Ljung chi-square test statistics for white noise residuals
  - autocorrelation function of residuals
  - partial autocorrelation function of residuals
  - inverse autocorrelation function of residuals
  - automatic outlier detection

Vector Time Series Analysis

The VARMAX procedure enables you to model both the dynamic relationship between the dependent variables and between the dependent and independent variables. The VARMAX procedure includes the following features:

- several modeling features:
  - vector autoregressive model
  - vector autoregressive model with exogenous variables

  - vector autoregressive and moving-average model
  - Bayesian vector autoregressive model
  - vector error correction model
  - Bayesian vector error correction model
  - GARCH-type multivariate conditional heteroscedasticity models

- criteria for automatically determining AR and MA orders:
  - Akaike Information Criterion (AIC)
  - Corrected AIC (AICC)
  - Hannan-Quinn (HQ) Criterion
  - Final Prediction Error (FPE)
  - Schwarz Bayesian Criterion (SBC), also known as Bayesian Information Criterion (BIC)

- AR order identification aids:
  - partial cross-correlations
  - Yule-Walker estimates
  - partial autoregressive coefficients
  - partial canonical correlations

- testing the presence of unit roots and cointegration:
  - Dickey-Fuller tests
  - Johansen cointegration test for nonstationary vector processes of integrated order one
  - Stock-Watson common trends test for the possibility of cointegration among nonstationary vector processes of integrated order one
  - Johansen cointegration test for nonstationary vector processes of integrated order two

- model parameter estimation methods:
  - Least Squares (LS)
  - Maximum Likelihood (ML)

- model checks and residual analysis using the following tests:
  - Durbin-Watson (DW) statistics
  - F test for autoregressive conditional heteroscedastic (ARCH) disturbance
  - F test for AR disturbance
  - Jarque-Bera normality test
  - Portmanteau test

- seasonal deterministic terms
- subset models
- multiple regression with distributed lags

- dead-start model that does not have present values of the exogenous variables
- Granger-causal relationships between two distinct groups of variables.
- infinite order AR representation
- impulse response function (or infinite order MA representation)
- decomposition of the predicted error covariances
- roots of the characteristic functions for both the AR and MA parts to evaluate the proximity of the roots to the unit circle
- contemporaneous relationships among the components of the vector time series
- forecasts

State Space Modeling and Forecasting

The STATESPACE procedure provides automatic model selection, parameter estimation, and forecasting of state space models. (*State space models* encompass an alternative general formulation of multivariate ARIMA models.) The STATESPACE procedure includes the following features:

- multivariate ARIMA modeling using the general state space representation of the stochastic process
- automatic model selection using Akaike's information criterion (AIC)
- user-specified state space models including restrictions
- transfer function models with random inputs
- any combination of simple and seasonal differencing; input series can be differenced to any order for any lag lengths
- forecasts with confidence limits
- can save selected and fitted model in a data set and reuse for forecasting
- wide range of output options; print any statistics concerning the data and their covariance structure, the model selection process, and the final model fit

Spectral Analysis

The SPECTRA procedure provides spectral analysis and cross-spectral analysis of time series. The SPECTRA procedure includes the following features:

- efficient calculation of periodogram and smoothed periodogram using fast finite Fourier transform and Chirp algorithms
- multiple spectral analysis, including raw and smoothed spectral and cross-spectral function estimates, with user-specified window weights
- choice of kernel for smoothing
- outputs the following spectral estimates to a SAS data set:
  - Fourier sine and cosine coefficients

  - periodogram
  - smoothed periodogram
  - cospectrum
  - quadrature spectrum
  - amplitude
  - phase spectrum
  - squared coherency

- Fisher's Kappa and Bartlett's Kolmogorov-Smirnov test statistic for testing a null hypothesis of white noise

Seasonal Adjustment

The X11 procedure provides seasonal adjustment of time series using the Census X-11 or X-11 ARIMA method. The X11 procedure is based on the U.S. Bureau of the Census X-11 seasonal adjustment program and also supports the X-11 ARIMA method developed by Statistics Canada. The X11 procedure includes the following features:

- decomposition of monthly or quarterly series into seasonal, trend, trading day, and irregular components
- both multiplicative and additive form of the decomposition
- includes all the features of the Census Bureau program
- supports the X-11 ARIMA method
- supports sliding spans analysis
- processes any number of variables at once with no maximum length for a series
- performs tests for stable, moving and combined seasonality
- can optionally print or store in SAS data sets the individual X11 tables showing the various components at different stages of the computation. Full control over what is printed or output
- can project seasonal component one year ahead enabling reintroduction of seasonal factors for an extrapolated series

The X12 procedure provides seasonal adjustment of time series using the X-12 ARIMA method. The X12 procedure is based on the U.S. Bureau of the Census X-12 ARIMA seasonal adjustment program (version 0.3) and also supports the X-11 ARIMA method developed by Statistics Canada and the previous X-11 method of the U.S. Census Bureau. The X12 procedure includes the following features:

- decomposition of monthly or quarterly series into seasonal, trend, trading day, and irregular components
- supports multiplicative, additive, pseudo-additive, and log additive forms of decomposition

- supports the X-12 ARIMA method
- supports regARIMA modeling
- automatically identifies outliers
- supports TRAMO-based automatic model selection
- uses regressors to process missing values within the span of the series
- processes any number of variables at once with no maximum length for a series
- performs tests for stable, moving and combined seasonality
- provides spectral analysis of original, seasonally adjusted, and irregular series
- optionally prints or stores in SAS a data set the individual X11 tables showing the various components at different stages of the decomposition. Offers full control over what is printed or output
- optionally projects seasonal component one year ahead, enabling reintroduction of seasonal factors for an extrapolated series

Structural Time Series Modeling and Forecasting

The UCM procedure provides a very flexible environment for analyzing time series data using Structural Time Series models, also called Unobserved Components Models (UCM). These models represent the observed series as a sum of suitably chosen components such as trend, seasonals, cycles, and regression effects. You can use the UCM procedure to formulate very comprehensive models that bring out all the salient features of the series under consideration. Structural models are applicable in the same situations where Box-Jenkins ARIMA models are applicable; however, the structural models tend to be more informative about the underlying stochastic structure of the series. The UCM procedure includes the following features:

- General Unobserved Components modeling where the models can include trend, multiple seasons and cycles, and regression effects
- Maximum likelihood estimation of the model parameters
- Model diagnostics that includes a variety of Goodness of Fit statistics, and extensive graphical diagnosis of the model residuals
- Forecasts and confidence limits for the series and all the model components
- Model-based seasonal decomposition
- Extensive plotting capability that includes:
  - Forecast and confidence interval plots for the series and model components such as trend, cycles, and seasons
  - Diagnostic plots such as residual plot, residual auto-correlation plots, etc.
  - Seasonal decomposition plots such as trend, trend plus cycles, trend plus cycles plus seasons, etc.
- Model-based interpolation of series missing values
- Full sample (also called smoothed) estimates of the model components

Time Series Cross-Sectional Regression Analysis

The TSCSREG procedure provides combined time series cross-sectional regression analysis. The TSCSREG procedure includes the following features:

- estimation of the regression parameters under several common error structures:
  - Fuller and Battese method (variance component model)
  - Parks method (autoregressive model)
  - Da Silva method (mixed variance component moving-average model)
  - one-way fixed effects
  - two-way fixed effects
  - one-way random effects
  - two-way random effects
- any number of model specifications
- unbalanced panel data for the fixed or random effects models
- variety of estimates and statistics including
  - underlying error components estimates
  - regression parameter estimates
  - standard errors of estimates
  - *t*-tests
  - R-squared statistic
  - correlation matrix of estimates
  - covariance matrix of estimates
  - autoregressive parameter estimate
  - cross-sectional components estimates
  - autocovariance estimates
  - F-tests of linear hypotheses about the regression parameters
  - specification tests

Automatic Time Series Forecasting

The FORECAST procedure provides forecasting of univariate time series using automatic trend extrapolation. PROC FORECAST is an easy-to-use procedure for automatic forecasting that uses simple popular methods that do not require statistical modeling of the time series, such as exponential smoothing, time trend with autoregressive errors, and the Holt-Winters method.

The FORECAST procedure supplements the powerful forecasting capabilities of the econometric and time series analysis procedures described above. You can use PROC FORECAST when you have many series to forecast and want to extrapolate trends without developing a model for each series.

The FORECAST procedure includes the following features:

- choice of the following four forecasting methods:
  - exponential smoothing: single, double, triple, or Holt two-parameter smoothing
  - stepwise autoregressive models with constant, linear, or quadratic trend and autoregressive errors to any order
  - Holt-Winters forecasting method with constant, linear, or quadratic trend
  - additive variant of the Holt-Winters method
- support for up to three levels of seasonality for Holt-Winters method: time-of-year, day-of-week, or time-of-day
- ability to forecast any number of variables at once
- forecast confidence limits for all methods

Time Series Interpolation and Frequency Conversion

The EXPAND procedure provides time interval conversion and missing value interpolation for time series. The EXPAND procedure includes the following features:

- conversion of time series frequency; for example, constructing quarterly estimates from annual series or aggregating quarterly values to annual values
- conversion of irregular observations to periodic observations
- interpolation of missing values in time series
- conversion of observation types; for example, estimate stocks from flows and vice versa. All possible conversions supported between
  - beginning of period
  - end of period
  - period midpoint
  - period total
  - period average
- conversion of time series phase shift; for example, conversion between fiscal years and calendar years
- choice of four interpolation methods:
  - cubic splines
  - linear splines
  - step functions
  - simple aggregation
- ability to transform series before and after interpolation (or without interpolation) using:
  - constant shift or scale
  - sign change or absolute value

  - logarithm, exponential, square root, square, logistic, inverse logistic
  - lags, leads, differences
  - classical decomposition
  - bounds, trims, reverse series
  - centered moving, cumulative, or backward moving average
  - centered moving, cumulative, or backward moving corrected sum of squares
  - centered moving, cumulative, or backward moving sum
  - centered moving, cumulative, or backward moving median
  - centered moving, cumulative, or backward moving variance

- support for a wide range of time series frequencies:
  - YEAR
  - SEMIYEAR
  - QUARTER
  - MONTH
  - SEMIMONTH
  - TENDAY
  - WEEK
  - WEEKDAY
  - DAY
  - HOUR
  - MINUTE
  - SECOND
- The basic interval types can be repeated or shifted to define a great variety of different frequencies, such as fiscal years, biennial periods, work shifts, and so forth.

Access to Financial and Economic Databases

The DATASOURCE procedure provides a convenient way to read time series data from data files supplied by a variety of different commercial and governmental data vendors. The DATASOURCE procedure includes the following features:

- support for data files distributed by the following data vendors:
  - DRI/McGraw-Hill
  - FAME Information Services
  - Haver Analytics
  - Standard & Poors Compustat Service
  - Center for Research in Security Prices (CRSP)
  - International Monetary Fund
  - U.S. Bureau of Labor Statistics

  - U.S. Bureau of Economic Analysis
  - Organization for Economic Cooperation and Development (OECD)

- ability to select the series, time range, and cross sections of data extracted
- can create an output data set containing descriptive information on the series available in the data file
- can read EBCDIC tapes on ASCII systems and vice versa

The SASECRSP interface engine enables random access to time series data residing in CRSPAccess database files and provides a seamless interface between CRSP and SAS data processing. The SASECRSP engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the CRSPAccess database, and how you would like to perform selection on the CRSP set you choose to access. The following data sets are available:

STKHEAD	header identification and summary data
NAMES	history array
SHARES	outstanding observation array
DELIST	delisting history array
PRC	Price or Bid/Ask Average Time Series
RET	Returns Time Series
BID, ASK, RETX	Returns without Dividends Time Series
SPREAD	Spread Between Bid and Ask Time Series
VOL	Volume Time Series
NUMTRD	Number of Trades Time Series
ALTPRCDT	Alternate Price Date Time Series
PORT1-PORT9	nine types of Portfolio Assignments and Portfolio Statistics.

The SASEFAME interface engine provides SAS and FAME users flexibility in accessing and processing time series data residing in either a FAME database or a SAS data set, and provides a seamless interface between FAME and SAS data processing. The SASEFAME engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the FAME database, and how you would like to convert the selected time series to the same time scale. The SAS DATA step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time. If you are running FAME in a client/server environment and have FAME CHLI capability on your FAME server, you can access your FAME remote data by specifying the port number of the TCP/IP service that is defined for your *frdb_m* and the node name of your FAME master server in your physical path.

The SASEHAVR interface engine is experimental for V9 and enables Windows users random access to economic and financial data residing in a HAVER ANALYTICS

Data Link Express (DLX) database. You can limit the range of data that is read from the time series and specify a desired conversion frequency. Start dates are recommended on the libname statement to help you save resources when processing large databases or when processing a large number of observations. You can further subsetting of your data by using the WHERE, KEEP, or DROP statements in your DATA step. You can use the SQL procedure to create a view of your resulting SAS data set.

Spreadsheet Calculations and Financial Report Generation

The COMPUTAB procedure generates tabular reports using a programmable data table.

The COMPUTAB procedure is especially useful when you need both the power of a programmable spreadsheet and a report generation system, and you want to set up a program to run in batch mode and generate routine reports. The COMPUTAB procedure includes the following features:

- report generation facility for creating tabular reports such as income statements, balance sheets, and other row and column reports for analyzing business or time series data
- can tailor report format to almost any desired specification
- uses the SAS programming language to provide complete control of the calculation and format of each item of the report
- reports definition in terms of a data table on which programming statements operate
- a single reference to a row or column brings the entire row or column into a calculation
- can create new rows and columns (such as totals, subtotals, and ratios) with a single programming statement
- access to individual table values is available when needed
- built-in features to provide consolidation reports over summarization variables

An alternate to the COMPUTAB procedure is the experimental SYLK procedure available in Base SAS. The documentation for the SYLK procedure can be found at http://support.sas.com/documentation/onlinedoc by selecting “Base SAS” from the Product-Specific Documentation list.

Loan Analysis, Comparison, and Amortization

The LOAN procedure provides analysis and comparison of mortgages and other installment loans. The LOAN procedure includes the following features:

- contract terms for any number of different loans may be input and various financing alternatives may be analyzed and compared
- analysis of four different types of loan contracts including

  - fixed rate
  - adjustable rate
  - buydown rate
  - balloon payment

- full control over adjustment terms for adjustable rate loans: life caps, adjustment frequency, and maximum and minimum rates
- support for a wide variety of payment and compounding intervals
- loan calculations can incorporate initialization costs, discount points, down payments, and prepayments (uniform or lump-sum)
- analysis of different rate adjustment scenarios for variable rate loans including
  - worst case
  - best case
  - fixed rate case
  - estimated case
- can make loan comparisons at different points in time
- can make loan comparisons at each analysis date on the basis of five different economic criteria
  - present worth of cost (net present value of all payments to date)
  - true interest rate (internal rate of return to date)
  - current periodic payment
  - total interest paid to date
  - outstanding balance
- can base loan comparisons on either after-tax or before-tax analysis
- reports best alternative when loans of equal amount are compared
- amortization schedules for each loan contract
- when starting date is specified, output shows payment dates rather than just payment sequence numbers
- can optionally print or output to SAS data sets the amortization schedules, loan summaries, and loan comparison information
- can specify rounding of payments to any number of decimal places

Time Series Forecasting System

SAS/ETS software includes the Time Series Forecasting System, a point-and-click application for exploring and analyzing univariate time series data. You can use the automatic model selection facility to select the best-fitting model for each time series, or you can use the system's diagnostic features and time series modeling tools interactively to develop forecasting models customized to best predict your time series. The system provides both graphical and statistical features to help you choose the best forecasting method for each series.

The system can be invoked from the Solutions menu under Analysis, by the Forecast command, and by the Forecasting icon in the Data Analysis folder of the SAS Desktop.

The following is a brief summary of the features of the Time Series Forecasting system. With the system you can

- use a wide variety of forecasting methods, including several kinds of exponential smoothing models, Winters method, and ARIMA (Box-Jenkins) models. You can also produce forecasts by combining the forecasts from several models.
- use predictor variables in forecasting models. Forecasting models can include time trend curves, regressors, intervention effects (dummy variables), adjustments you specify, and dynamic regression (transfer function) models.
- view plots of the data, predicted versus actual values, prediction errors, and forecasts with confidence limits. You can plot changes or transformations of series, zoom in on parts of the graphs, or plot autocorrelations.
- use hold-out samples to select the best forecasting method.
- compare goodness-of-fit measures for any two forecasting models side by side or list all models sorted by a particular fit statistic.
- view the predictions and errors for each model in a spreadsheet or view and compare the forecasts from any two models in a spreadsheet.
- examine the fitted parameters of each forecasting model and their statistical significance.
- control the automatic model selection process: the set of forecasting models considered, the goodness-of-fit measure used to select the best model, and the time period used to fit and evaluate models.
- customize the system by adding forecasting models for the automatic model selection process and for point-and-click manual selection.
- save your work in a project catalog.
- print an audit trail of the forecasting process.
- save and print system output including spreadsheets and graphs.

Investment Analysis System

The Investment Analysis System is an interactive environment for the time-value of money of a variety of investments:

- Loans
- Savings
- Depreciations
- Bonds
- Generic cashflows

Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

These analyses can help answer a number of questions you may have about your investments:

- Which option is more profitable or less costly?
- Is it better to buy or rent?
- Are the extra fees for refinancing at a lower interest rate justified?
- What is the balance of this account after saving this amount periodically for so many years?
- How much is legally tax-deductible?
- Is this a reasonable price?

Investment Analysis can be beneficial to users in many industries for a variety of decisions:

- manufacturing: cost justification of automation or any capital investment, replacement analysis of major equipment, or economic comparison of alternative designs
- government: setting funds for services
- finance: investment analysis and portfolio management for fixed-income securities

Related SAS Software

Many features not found in SAS/ETS software are available in other parts of the SAS System. If you do not find something you need in SAS/ETS software, you may find it in one of the following SAS software products.

Base SAS Software

The features provided by SAS/ETS software are extensions to the features provided by Base SAS software. Many data management and reporting capabilities you will need are part of Base SAS software. Refer to *SAS Language: Reference* and the *SAS Procedures Guide* for documentation of Base SAS software.

The following sections summarize Base SAS software features of interest to users of SAS/ETS software. See Chapter 2 for further discussion of some of these topics as they relate to time series data and SAS/ETS software.

SAS DATA Step

The DATA step is your primary tool for reading and processing data in the SAS System. The DATA step provides a powerful general purpose programming language that enables you to perform all kinds of data processing tasks. The DATA step is documented in *SAS Language: Reference*.

Base SAS Procedures

Base SAS software includes many useful SAS procedures. Base SAS procedures are documented in the *SAS Procedures Guide*. The following is a list of Base SAS procedures you may find useful:

CATALOG	for managing SAS catalogs
CHART	for printing charts and histograms
COMPARE	for comparing SAS data sets
CONTENTS	for displaying the contents of SAS data sets
COPY	for copying SAS data sets
CORR	for computing correlations
CPORT	for moving SAS data libraries between computer systems
DATASETS	for deleting or renaming SAS data sets
FREQ	for computing frequency crosstabulations
MEANS	for computing descriptive statistics and summarizing or collapsing data over cross sections
PLOT	for printing scatter plots
PRINT	for printing SAS data sets
RANK	for computing rankings or order statistics
SORT	for sorting SAS data sets
SQL	for processing SAS data sets with Structured Query Language
STANDARD	for standardizing variables to a fixed mean and variance
SYLK	for translating spreadsheets to batch SAS programs. The SYLK procedure is experimental. The documentation can be found at http://support.sas.com/documentation/onlinedoc by selecting “Base SAS” from the Product-Specific Documentation list

TABULATE	for printing descriptive statistics in tabular format
TIMEPLOT	for plotting variables over time
TRANSPOSE	for transposing SAS data sets
UNIVARIATE	for computing descriptive statistics

Global Statements

Global statements can be specified anywhere in your SAS program, and they remain in effect until changed. Global statements are documented in *SAS Language: Reference*. You may find the following SAS global statements useful:

FILENAME	for accessing data files
FOOTNOTE	for printing footnote lines at the bottom of each page
%INCLUDE	for including files of SAS statements
LIBNAME	for accessing SAS data libraries
OPTIONS	for setting various SAS system options
RUN	for executing the preceding SAS statements
TITLE	for printing title lines at the top of each page
X	for issuing host operating system commands from within your SAS session

Some Base SAS statements can be used with any SAS procedure, including SAS/ETS procedures. These statements are not global, and they only affect the SAS procedure they are used with. These statements are documented in *SAS Language: Reference*.

The following Base SAS statements are useful with SAS/ETS procedures:

BY	for computing separate analyses for groups of observations
FORMAT	for assigning formats to variables
LABEL	for assigning descriptive labels to variables
WHERE	for subsetting data to restrict the range of data processed or to select or exclude observations from the analysis

SAS Functions

SAS functions can be used in DATA step programs and in the COMPUTAB and MODEL procedures. The following kinds of functions are available:

- character functions for manipulating character strings
- date and time functions, for performing date and calendar calculations
- financial functions, for performing financial calculations such as depreciation, net present value, periodic savings, and internal rate of return

- lagging and differencing functions, for computing lags and differences
- mathematical functions, for computing data transformations and other mathematical calculations
- probability functions, for computing quantiles of statistical distributions and the significance of test statistics
- random number functions, for simulation experiments
- sample statistics functions, for computing means, standard deviations, kurtosis, and so forth

SAS functions are documented in *SAS Language: Reference*. Chapter 2, "Working with Time Series Data," discusses the use of date and time and lagging and differencing functions. Chapter 3, "Date Intervals, Formats, and Functions," contains a reference list of date and time functions. Chapter 4, "SAS Macros and Functions," documents more financial functions that are not listed in *SAS Language: Reference*.

Formats, Informats, and Time Intervals

Base SAS software provides formats to control the printing of data values, informats to read data values, and time intervals to define the frequency of time series. See Chapter 3, "Date Intervals, Formats, and Functions," for more information.

SAS/GRAPH Software

SAS/GRAPH software includes procedures that create two- and three-dimensional high resolution color graphics plots and charts. You can generate output that graphs the relationship of data values to one another, enhance existing graphs, or simply create graphics output that is not tied to data. SAS/GRAPH software can produce

- charts
- plots
- maps
- text
- three-dimensional graphs

With SAS/GRAPH software you can produce high-resolution color graphics plots of time series data.

SAS/STAT Software

SAS/STAT software is of interest to users of SAS/ETS software because many econometric and other statistical methods not included in SAS/ETS software are provided in SAS/STAT software.

SAS/STAT software includes procedures for a wide range of statistical methodologies including

- logistic regression
- censored regression
- principal component analysis
- structural equation models using covariance structure analysis
- factor analysis
- survival analysis
- discriminant analysis
- cluster analysis
- categorical data analysis; log-linear and conditional logistic models
- general linear models
- mixed linear and nonlinear models
- generalized linear models
- response surface analysis
- kernel density estimation
- LOESS regression
- spline regression
- two-dimensional kriging
- multiple imputation for missing values

SAS/IML Software

SAS/IML software gives you access to a powerful and flexible programming language (Interactive Matrix Language) in a dynamic, interactive environment. The fundamental object of the language is a data matrix. You can use SAS/IML software interactively (at the statement level) to see results immediately, or you can store statements in a module and execute them later. The programming is dynamic because necessary activities such as memory allocation and dimensioning of matrices are done automatically.

You can access built-in operators and call routines to perform complex tasks such as matrix inversion or eigenvector generation. You can define your own functions and subroutines using SAS/IML modules. You can perform operations on an entire data matrix. You have access to a wide choice of data management commands. You can read, create, and update SAS data sets from inside SAS/IML software without ever using the DATA step.

SAS/IML software is of interest to users of SAS/ETS software because it enables you to program your own econometric and time series methods in the SAS System. It contains subroutines for time series operators and for general function optimization. If you need to perform a statistical calculation not provided as an automated feature by SAS/ETS or other SAS software, you can use SAS/IML software to program the matrix equations for the calculation.

Kalman Filtering and Time Series Analysis in SAS/IML

SAS/IML software includes a library for Kalman filtering and time series analysis which provides the following functions:

- generating univariate, multivariate, and fractional time series
- computing likelihood function of ARMA, VARMA, and ARFIMA models
- computing an autocovariance function of ARMA, VARMA, and ARFIMA models
- checking the stationarity of ARMA and VARMA models
- filtering and smoothing of time series models using Kalman method
- fitting AR, periodic AR, time-varying coefficient AR, VAR, and ARFIMA models
- handling Bayesian seasonal adjustment model

Refer to Chapter 10, "Time Series Analysis and Examples," (*SAS/IML User's Guide*) for details.

SAS/INSIGHT Software

SAS/INSIGHT software is a highly interactive tool for data analysis. You can explore data through a variety of interactive graphs including bar charts, scatter plots, box plots, and three-dimensional rotating plots. You can examine distributions and perform parametric and nonparametric regression, analyze general linear models and generalized linear models, examine correlation matrixes, and perform principal component analyses. Any changes you make to your data show immediately in all graphs and analyses. You can also configure SAS/INSIGHT software to produce graphs and analyses tailored to the way you work.

SAS/INSIGHT software is an integral part of the SAS System. You can use it to examine output from a SAS procedure, and you can use any SAS procedure to analyze results from SAS/INSIGHT software.

SAS/INSIGHT software includes features for both displaying and analyzing data interactively. A data window displays a SAS data set as a table with columns of the table displaying variables and rows displaying observations. Data windows provide data management features for editing, transforming, subsetting, and sorting data. A graph window displays different types of graphs: bar charts, scatter plots, box plots, and rotating plots. Graph windows provide interactive exploratory techniques such as data brushing and highlighting. Analysis windows display statistical analyses in the form of graphs and tables. Analysis window features include

- univariate statistics
- robust estimates
- density estimates
- cumulative distribution functions

- theoretical quantile-quantile plots
- multiple regression analysis with numerous diagnostic capabilities
- general linear models
- generalized linear models
- smoothing spline estimates
- kernel density estimates
- correlations
- principal components

SAS/INSIGHT software may be of interest to users of SAS/ETS software for interactive graphical viewing of data, editing data, exploratory data analysis, and checking distributional assumptions.

SAS/OR Software

SAS/OR software provides SAS procedures for operations research and project planning and includes a menu driven system for project management. SAS/OR software has features for

- solving transportation problems
- linear, integer, and mixed-integer programming
- nonlinear programming and optimization
- scheduling projects
- plotting Gantt charts
- drawing network diagrams
- solving optimal assignment problems
- network flow programming

SAS/OR software may be of interest to users of SAS/ETS software for its mathematical programming features. In particular, the NLP procedure in SAS/OR software solves nonlinear programming problems and can be used for constrained and unconstrained maximization of user-defined likelihood functions.

SAS/QC Software

SAS/QC software provides a variety of procedures for statistical quality control and quality improvement. SAS/QC software includes procedures for

- Shewhart control charts
- cumulative sum control charts
- moving average control charts

- process capability analysis
- Ishikawa diagrams
- Pareto charts
- experimental design

SAS/QC software also includes the SQC menu system for interactive application of statistical quality control methods and the ADX Interface for experimental design.

MLE for User-Defined Likelihood Functions

There are three SAS procedures that enable you to do maximum likelihood estimation of parameters in an arbitrary model with a likelihood function that you define: PROC MODEL, PROC NLP, and PROC IML.

The MODEL procedure in SAS/ETS software enables you to minimize general log-likelihood functions for the error term of a model.

The NLP procedure in SAS/OR software is a general nonlinear programming procedure that can maximize a general function subject to linear equality or inequality constraints. You can use PROC NLP to maximize a user-defined nonlinear likelihood function.

You can use the IML procedure in SAS/IML software for maximum likelihood problems. The optimization routines used by PROC NLP are available through IML subroutines. You can write the likelihood function in the SAS/IML matrix language and call the constrained and unconstrained nonlinear programming subroutines to maximize the likelihood function with respect to the parameter vector.

Other Statistical Tools

Many other statistical tools are available in Base SAS, SAS/STAT, SAS/OR, SAS/QC, SAS/INSIGHT, and SAS/IML software. If you don't find something you need in SAS/ETS software, you may find it in SAS/STAT software and in Base SAS software. If you still don't find it, look in other SAS software products or contact the SAS Institute Technical Support staff.

References

Amal, S. and Weselowski, R. (1993), “Practical Econometric Analysis for Assessment of Real Property: Using the SAS System on Personal Computers,” *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 385-390. Cary, NC: SAS Institute Inc.

Benseman, B. (1990), “Better Forecasting with SAS/ETS Software,” *Proceedings of the Fifteenth Annual SAS Users Group International Conference*, 494-497. Cary, NC: SAS Institute Inc.

Calise, A. and Earley, J. (1997), “Forecasting College Enrollment Using the SAS System,” *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, 1326-1329. Cary, NC: SAS Institute Inc.

Early, J., Sweeney, J., and Zekavat, S.M. (1989), “PROC ARIMA and the Dow Jones Stock Index,” *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 371-375. Cary, NC: SAS Institute Inc.

Fischetti, T., Heathcote, S. and Perry, D. (1993), “Using SAS to Create a Modular Forecasting System,” *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 580-585. Cary, NC: SAS Institute Inc.

Fleming, N.S., Gibson, E. and Fleming, D.G. (1996), “The Use of PROC ARIMA to Test an Intervention Effect,” *Proceedings of the Twenty-First Annual SAS Users Group International Conference*, 1317-1326. Cary, NC: SAS Institute Inc.

Hisnanick, J.J. (1991), “Evaluating Input Separability in a Model of of the U.S. Manufacturing Sector,” *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 688-693. Cary, NC: SAS Institute Inc.

Hisnanick, J.J. (1992), “Using PROC ARIMA in Forecasting the Demand and Utilization of Inpatient Hospital Services,” *Proceedings of the Seventeenth Annual SAS Users Group International Conference*, 383-391. Cary, NC: SAS Institute Inc.

Hisnanick, J.J. (1993), “Using SAS/ETS in Applied Econometrics: Parameters Estimates for the CES-Translog Specification,” *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 275-279. Cary, NC: SAS Institute Inc.

Hoyer, K.K. and Gross, K.C. (1993), “Spectral Decomposition and Reconstruction of Nuclear Plant Signals,” *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 1153-1158. Cary, NC: SAS Institute Inc.

Keshani, D.A. and Taylor, T.N. (1992), “Weather Sensitive Appliance Load Curves; Conditional Demand Estimation,” *Proceedings of the Annual SAS Users Group International Conference*, 422-430. Cary, NC: SAS Institute Inc.

Khan, M.H. (1990), “Transfer Function Model for Gloss Prediction of Coated Aluminum Using the ARIMA Procedure,” *Proceedings of the Fifteenth Annual SAS Users Group International Conference*, 517-522. Cary, NC: SAS Institute Inc.

Le Bouton, K.J. (1989), "Performance Function for Aircraft Production Using PROC SYSLIN and L^2 Norm Estimation," *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 424-426. Cary, NC: SAS Institute Inc.

Lin, L. and Myers, S.C. (1988), "Forecasting the Economy using the Composite Leading Index, Its Components, and a Rational Expectations Alternative," *Proceedings of the Thirteenth Annual SAS Users Group International Conference*, 181-186. Cary, NC: SAS Institute Inc.

McCarty, L. (1994), "Forecasting Operational Indices Using SAS/ETS Software," *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 844-848. Cary, NC: SAS Institute Inc.

Morelock, M.M., Pargellis, C.A., Graham, E.T., Lamarre, D., and Jung, G. (1995), "Time-Resolved Ligand Exchange Reactions: Kinetic Models for competitive Inhibitors with Recombinant Human Renin," *Journal of Medical Chemistry*, 38, 1751–1761.

Parresol, B.R. and Thomas, C.E. (1991), "Econometric Modeling of Sweetgum Stem Biomass Using the IML and SYSLIN Procedures," *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 694-699. Cary, NC: SAS Institute Inc.

SAS institute Inc. (1993), *SAS/ETS Software: Applications Guide 1*, Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS institute Inc. (1993), *SAS/ETS Software: Applications Guide 2*, Version 6, First Edition, Cary, NC: SAS Institute Inc.

Chapter 2
Working with Time Series Data

Chapter Contents

Chapter 2
Working with Time Series Data

Overview

This chapter discusses working with time series data in the SAS System. The following topics are included:

- dating time series and working with SAS date and datetime values
- subsetting data and selecting observations
- storing time series data in SAS data sets
- specifying time series periodicity and time intervals
- plotting time series
- using calendar and time interval functions
- computing lags and other functions across time
- transforming time series
- transposing time series data sets
- interpolating time series
- reading time series data recorded in different ways

In general, this chapter focuses on using features of the SAS programming language and not on features of SAS/ETS software. However, since SAS/ETS procedures are used to analyze time series, understanding how to use the SAS programming language to work with time series data is important for the effective use of SAS/ETS software.

You do not need to read this chapter to use SAS/ETS procedures. If you are already familiar with SAS programming you may want to skip this chapter, or you may refer to sections of this chapter for help on specific time series data processing questions.

Time Series and SAS Data Sets

Introduction

To analyze data with the SAS System, data values must be stored in a SAS data set. A SAS data set is a matrix or table of data values organized into variables and observations.

The *variables* in a SAS data set label the columns of the data matrix and the observations in a SAS data set are the rows of the data matrix. You can also think of a SAS data set as a kind of file, with the observations representing records in the file and the variables representing fields in the records. (Refer to *SAS Language: Reference, Version 6* for more information about SAS data sets.)

Usually, each observation represents the measurement of one or more variables for the individual subject or item observed. Often, the values of some of the variables in the data set are used to identify the individual subjects or items that the observations measure. These identifying variables are referred to as *ID variables*.

For many kinds of statistical analysis, only relationships among the variables are of interest, and the identity of the observations does not matter. ID variables may not be relevant in such a case.

However, for time series data the identity and order of the observations are crucial. A time series is a set of observations made at a succession of equally spaced points in time.

For example, if the data are monthly sales of a company's product, the variable measured is sales of the product and the thing observed is the operation of the company during each month. These observations can be identified by year and month. If the data are quarterly gross national product, the variable measured is final goods production and the thing observed is the economy during each quarter. These observations can be identified by year and quarter.

For time series data, the observations are identified and related to each other by their position in time. Since the SAS system does not assume any particular structure to the observations in a SAS data set, there are some special considerations needed when storing time series in a SAS data set.

The main considerations are how to associate dates with the observations and how to structure the data set so that SAS/ETS procedures and other SAS procedures will recognize the observations of the data set as constituting time series. These issues are discussed in following sections.

Reading a Simple Time Series

Time series data can be recorded in many different ways. The section "Reading Time Series Data" later in this chapter discusses some of the possibilities. The example below shows a simple case.

The following SAS statements read monthly values of the U.S. Consumer Price Index for June 1990 through July 1991. The data set USCPI is shown in Figure 2.1.

```
data uscpi;
   input year month cpi;
datalines;
1990  6 129.9
1990  7 130.4
1990  8 131.6
1990  9 132.7
1990 10 133.5
1990 11 133.8
1990 12 133.8
```

```
1991  1 134.6
1991  2 134.8
1991  3 135.0
1991  4 135.2
1991  5 135.6
1991  6 136.0
1991  7 136.2
;

proc print data=uscpi;
run;
```

```
                     Obs    year    month     cpi

                       1    1990       6     129.9
                       2    1990       7     130.4
                       3    1990       8     131.6
                       4    1990       9     132.7
                       5    1990      10     133.5
                       6    1990      11     133.8
                       7    1990      12     133.8
                       8    1991       1     134.6
                       9    1991       2     134.8
                      10    1991       3     135.0
                      11    1991       4     135.2
                      12    1991       5     135.6
                      13    1991       6     136.0
                      14    1991       7     136.2
```

Figure 2.1. Time Series Data

When a time series is stored in the manner shown by this example, the terms *series* and *variable* can be used interchangeably. There is one observation per row, and one series/variable per column.

Dating Observations

The SAS System supports special date, datetime, and time values, which make it easy to represent dates, perform calendar calculations, and identify the time period of observations in a data set.

The preceding example used the ID variables YEAR and MONTH to identify the time periods of the observations. For a quarterly data set, you might use YEAR and QTR as ID variables. A daily data set might have the ID variables YEAR, MONTH, and DAY. Clearly, it would be more convenient to have a single ID variable that could be used to identify the time period of observations, regardless of their frequency.

The following section, "SAS Date, Datetime, and Time Values," discusses how the SAS System represents dates and times internally and how to specify date, datetime, and time values in a SAS program. The section "Reading Date and Datetime Values with Informats" discusses how to control the display of date and datetime values in SAS output and how to read in date and time values from data records. Later sections discuss other issues concerning date and datetime values, specifying time intervals, data periodicity, and calendar calculations.

SAS date and datetime values and the other features discussed in the following sections are also described in *SAS Language: Reference*. Reference documentation on these features is also provided in Chapter 3, "Date Intervals, Formats, and Functions."

SAS Date, Datetime, and Time Values

Year 2000 Compliance

SAS software correctly represents dates from 1582 AD to the year 20,000 AD. If dates in an external data source are represented with four-digit-year values SAS can read, write and compute these dates. If the dates in an external data source are two-digit years, SAS software provides informats, functions, and formats to read, manipulate, and output dates that are Year 2000 compliant. The YEARCUTOFF= system option can also be used to interpret dates with two-digit years by specifying the first year of a 100-year span that will be used in informats and functions. The default value for the YEARCUTOFF= option is 1920.

SAS Date Values

The SAS System represents dates as the number of days since a reference date. The reference date, or date zero, used for SAS date values is 1 January 1960. Thus, for example, 3 February 1960 is represented by the SAS System as 33. The SAS date for 17 October 1991 is 11612.

Dates represented in this way are called SAS *date values*. Any numeric variable in a SAS data set whose values represent dates in this way is called a SAS *date variable*.

Representing dates as the number of days from a reference date makes it easy for the computer to store them and perform calendar calculations, but these numbers are not meaningful to users. However, you never have to use SAS date values directly, since SAS automatically converts between this internal representation and ordinary ways of expressing dates, provided that you indicate the format with which you want the date values to be displayed. (Formatting of date values is explained in a following section.)

SAS Date Constants

SAS date values are written in a SAS program by placing the dates in single quotes followed by a D. The date is represented by the day of the month, the three letter abbreviation of the month name, and the year.

For example, SAS reads the value '17OCT1991'D the same as 11612, the SAS date value for 17 October 1991. Thus, the following SAS statements print DATE=11612.

```
data _null_;
  date = '17oct1991'd;
  put date=;
run;
```

The year value can be given with two or four digits, so '17OCT91'D is the same as '17OCT1991'D. (The century assumed for a two-digit year value can be controlled with the YEARCUTOFF= option in the OPTIONS statement. Refer to the *SAS Language: Reference* for information on YEARCUTOFF=.)

SAS Datetime Values and Datetime Constants

To represent both the time of day and the date, the SAS System uses *datetime values*. SAS datetime values represent the date and time as the number of seconds the time is from a reference time. The reference time, or time zero, used for SAS datetime values is midnight, 1 January 1960. Thus, for example, the SAS datetime value for 17 October 1991 at 2:45 in the afternoon is 1003329900.

To specify datetime constants in a SAS program, write the date and time in single quotes followed by DT. To write the date and time in a SAS datetime constant, write the date part using the same syntax as for date constants, and follow the date part with the hours, the minutes, and the seconds, separating the parts with colons. The seconds are optional.

For example, in a SAS program you would write 17 October 1991 at 2:45 in the afternoon as '17OCT91:14:45'DT. SAS reads this as 1003329900. Table 2.1 shows some other examples of datetime constants.

Table 2.1. Examples of Datetime Constants

Datetime Constant	Time
'17OCT1991:14:45:32'DT	32 seconds past 2:45 p.m., 17 October 1991
'17OCT1991:12:5'DT	12:05 p.m., 17 October 1991
'17OCT1991:2:0'DT	2 AM, 17 October 1991
'17OCT1991:0:0'DT	midnight, 17 October 1991

SAS Time Values

The SAS System also supports *time values*. SAS time values are just like datetime values, except that the date part is not given. To write a time value in a SAS program, write the time the same as for a datetime constant but use T instead of DT. For example, 2:45:32 p.m. is written '14:45:32'T. Time values are represented by a number of seconds since midnight, so SAS reads '14:45:32'T as 53132.

SAS time values are not very useful for identifying time series, since usually both the date and the time of day are needed. Time values are not discussed further in this book.

Reading Date and Datetime Values with Informats

The SAS System provides a selection of *informats* for reading SAS date and datetime values from date and time values recorded in ordinary notations.

A SAS informat is an instruction that converts the values from a character string representation into the internal numerical value of a SAS variable. Date informats convert dates from ordinary notations used to enter them to SAS date values; datetime informats convert date and time from ordinary notation to SAS datetime values.

For example, the following SAS statements read monthly values of the U.S. Consumer Price Index. Since the data are monthly, you could identify the date with the variables YEAR and MONTH, as in the previous example. Instead, in this example the time periods are coded as a three-letter month abbreviation followed by the year. The informat MONYY. is used to read month-year dates coded this way and to express them as SAS date values for the first day of the month, as follows.

```
data uscpi;
   input date: monyy7. cpi;
datalines;
jun1990 129.9
jul1990 130.4
aug1990 131.6
sep1990 132.7
oct1990 133.5
nov1990 133.8
dec1990 133.8
jan1991 134.6
feb1991 134.8
mar1991 135.0
apr1991 135.2
may1991 135.6
jun1991 136.0
jul1991 136.2
;
```

The SAS System provides informats for most common notations for dates and times. See Chapter 3 for more information on the date and datetime informats available.

Formatting Date and Datetime Values

The SAS System provides *formats* to convert the internal representation of date and datetime values used by SAS to ordinary notations for dates and times. Several different formats are available for displaying dates and datetime values in most of the commonly used notations.

A SAS format is an instruction that converts the internal numerical value of a SAS variable to a character string that can be printed or displayed. Date formats convert SAS date values to a readable form; datetime formats convert SAS datetime values to a readable form.

In the preceding example, the variable DATE was set to the SAS date value for the first day of the month for each observation. If the data set USCPI were printed or otherwise displayed, the values shown for DATE would be the number of days since 1 January 1960. (See the "DATE with no format" column in Figure 2.2.) To display date values appropriately, use the FORMAT statement.

The following example processes the data set USCPI to make several copies of the variable DATE and uses a FORMAT statement to give different formats to these copies. The format cases shown are the MONYY7. format (for the DATE variable), the DATE9. format (for the DATE1 variable), and no format (for the DATE0 variable). The PROC PRINT output in Figure 2.2 shows the effect of the different formats on how the date values are printed.

```
data fmttest;
   set uscpi;
   date0 = date;
   date1 = date;
   label date  = "DATE with MONYY7. format"
         date1 = "DATE with DATE9. format"
         date0 = "DATE with no format";
   format date monyy7. date1 date9.;
run;

proc print data=fmttest label;
run;
```

```
                 DATE with                                DATE with
                  MONYY.                     DATE with      DATE.
          Obs     format         cpi         no format     format

            1     JUN1990       129.9          11109      01JUN1990
            2     JUL1990       130.4          11139      01JUL1990
            3     AUG1990       131.6          11170      01AUG1990
            4     SEP1990       132.7          11201      01SEP1990
            5     OCT1990       133.5          11231      01OCT1990
            6     NOV1990       133.8          11262      01NOV1990
            7     DEC1990       133.8          11292      01DEC1990
            8     JAN1991       134.6          11323      01JAN1991
            9     FEB1991       134.8          11354      01FEB1991
           10     MAR1991       135.0          11382      01MAR1991
```

Figure 2.2. SAS Date Values Printed with Different Formats

The appropriate format to use for SAS date or datetime valued ID variables depends on the sampling frequency or periodicity of the time series. Table 2.2 shows recommended formats for common data sampling frequencies and shows how the date ’17OCT1991’D or the datetime value ’17OCT1991:14:45:32’DT is displayed by these formats.

Table 2.2. Formats for Different Sampling Frequencies

ID values	Periodicity	FORMAT	Example
SAS Date	Annual	YEAR4.	1991
	Quarterly	YYQC6.	1991:4
	Monthly	MONYY7.	OCT1991
	Weekly	WEEKDATX23.	Thursday, 17 Oct 1991
		DATE9.	17OCT1991
	Daily	DATE9.	17OCT1991
SAS Datetime	Hourly	DATETIME10.	17OCT91:14
	Minutes	DATETIME13.	17OCT91:14:45
	Seconds	DATETIME16.	17OCT91:14:45:32

See Chapter 3 for more information on the date and datetime formats available.

The Variables DATE and DATETIME

SAS/ETS procedures enable you to identify time series observations in many different ways to suit your needs. As discussed in preceding sections, you can use a combination of several ID variables, such as YEAR and MONTH for monthly data.

However, using a single SAS date or datetime ID variable is more convenient and enables you to take advantage of some features SAS/ETS procedures provide for processing ID variables. One such feature is automatic extrapolation of the ID variable to identify forecast observations. These features are discussed in following sections.

Thus, it is a good practice to include a SAS date or datetime ID variable in all the time series SAS data sets you create. It is also a good practice to always give the date or datetime ID variable a format appropriate for the data periodicity.

You can name a SAS date or datetime valued ID variable any name conforming to SAS variable name requirements. However, you may find working with time series data in SAS easier and less confusing if you adopt the practice of always using the same name for the SAS date or datetime ID variable.

This book always names the dating ID variable "DATE" if it contains SAS date values or "DATETIME" if it contains SAS datetime values. This makes it easy to recognize the ID variable and also makes it easy to recognize whether this ID variable uses SAS date or datetime values.

Sorting by Time

Many SAS/ETS procedures assume the data are in chronological order. If the data are not in time order, you can use the SORT procedure to sort the data set. For example

```
proc sort data=a;
   by date;
run;
```

There are many ways of coding the time ID variable or variables, and some ways do not sort correctly. If you use SAS date or datetime ID values as suggested in the preceding section, you do not need to be concerned with this issue. But if you encode date values in nonstandard ways, you need to consider whether your ID variables will sort.

SAS date and datetime values always sort correctly, as do combinations of numeric variables like YEAR, MONTH, and DAY used together. Julian dates also sort correctly. (Julian dates are numbers of the form *yyddd*, where *yy* is the year and *ddd* is the day of the year. For example 17 October 1991 has the Julian date value 91290.)

Calendar dates such as numeric values coded as *mmddyy* or *ddmmyy* do not sort correctly. Character variables containing display values of dates, such as dates in the notation produced by SAS date formats, generally do not sort correctly.

Subsetting Data and Selecting Observations

It is often necessary to subset data for analysis. You may need to subset data to

- restrict the time range. For example, you want to perform a time series analysis using only recent data and ignoring observations from the distant past.
- select cross sections of the data. (See the section "Cross-sectional Dimensions and BY Groups" later in this chapter.) For example, you have a data set with observations over time for each of several states, and you want to analyze the data for a single state.
- select particular kinds of time series from an interleaved form data set. (See the section "Interleaved Time Series and the _TYPE_ Variable" later in this chapter.) For example, you have an output data set produced by the FORECAST procedure that contains both forecast and confidence limits observations, and you want to extract only the forecast observations.
- exclude particular observations. For example, you have an outlier in your time series, and you want to exclude this observation from the analysis.

You can subset data either by using the DATA step to create a subset data set or by using a WHERE statement with the SAS procedure that analyzes the data.

A typical WHERE statement used in a procedure has the form

```
proc arima data=full;
   where '31dec1993'd < day < '26mar1994'd;
   identify var=close;
run;
```

For complete reference documentation on the WHERE statement refer to *SAS Language: Reference*.

Subsetting SAS Data Sets

To create a subset data set, specify the name of the subset data set on the DATA statement, bring in the full data set with a SET statement, and specify the subsetting criteria with either subsetting IF statements or WHERE statements.

For example, suppose you have a data set containing time series observations for each of several states. The following DATA step uses a WHERE statement to exclude observations with dates before 1970 and uses a subsetting IF statement to select observations for the state NC:

```
data subset;
   set full;
   where date >= '1jan1970'd;
   if state = 'NC';
run;
```

In this case, it makes no difference logically whether the WHERE statement or the IF statement is used, and you can combine several conditions on one subsetting statement. The following statements produce the same results as the previous example:

```
data subset;
   set full;
   if date >= '1jan1970'd & state = 'NC';
run;
```

The WHERE statement acts on the input data sets specified in the SET statement before observations are processed by the DATA step program, whereas the IF statement is executed as part of the DATA step program. If the input data set is indexed, using the WHERE statement can be more efficient than using the IF statement. However, the WHERE statement can only refer to variables in the input data set, not to variables computed by the DATA step program.

To subset the variables of a data set, use KEEP or DROP statements or use KEEP= or DROP= data set options. Refer to *SAS Language: Reference* for information on KEEP and DROP statements and SAS data set options.

For example, suppose you want to subset the data set as in the preceding example, but you want to include in the subset data set only the variables DATE, X, and Y. You could use the following statements:

```
data subset;
   set full;
   if date >= '1jan1970'd & state = 'NC';
   keep date x y;
run;
```

Using the WHERE Statement with SAS Procedures

Use the WHERE statement with SAS procedures to process only a subset of the input data set. For example, suppose you have a data set containing monthly observations for each of several states, and you want to use the AUTOREG procedure to analyze data since 1970 for the state NC. You could use the following:

```
proc autoreg data=full;
   where date >= '1jan1970'd & state = 'NC';
  ... additional statements ...
run;
```

You can specify any number of conditions on the WHERE statement. For example, suppose that a strike created an outlier in May 1975, and you want to exclude that observation. You could use the following:

```
proc autoreg data=full;
   where date >= '1jan1970'd & state = 'NC'
       & date ^= '1may1975'd;
  ... additional statements ...
run;
```

Using SAS Data Set Options

You can use the OBS= and FIRSTOBS= data set options to subset the input data set. These options cannot be used in conjunction with the WHERE statement.

For example, the following statements print observations 20 through 25 of the data set FULL.

```
proc print data=full(firstobs=20 obs=25);
run;
```

You can use KEEP= and DROP= data set options to exclude variables from the input data set. Refer to *SAS Language: Reference* for information on SAS data set options.

Storing Time Series in a SAS Data Set

This section discusses aspects of storing time series in SAS data sets. The topics discussed are the standard form of a time series data set, storing several series with different time ranges in the same data set, omitted observations, cross-sectional dimensions and BY groups, and interleaved time series.

Any number of time series can be stored in a SAS data set. Normally, each time series is stored in a separate variable. For example, the following statements augment the USCPI data set read in the previous example with values for the producer price index.

```
data usprice;
   input date monyy7. cpi ppi;
   format date monyy7.;
   label cpi = "Consumer Price Index"
         ppi = "Producer Price Index";
datalines;
jun1990 129.9 114.3
jul1990 130.4 114.5
aug1990 131.6 116.5
sep1990 132.7 118.4
oct1990 133.5 120.8
nov1990 133.8 120.1
dec1990 133.8 118.7
jan1991 134.6 119.0
feb1991 134.8 117.2
mar1991 135.0 116.2
apr1991 135.2 116.0
may1991 135.6 116.5
jun1991 136.0 116.3
jul1991 136.2 116.0
;

proc print data=usprice;
run;
```

```
                    Obs      date      cpi      ppi

                      1    JUN1990    129.9    114.3
                      2    JUL1990    130.4    114.5
                      3    AUG1990    131.6    116.5
                      4    SEP1990    132.7    118.4
                      5    OCT1990    133.5    120.8
                      6    NOV1990    133.8    120.1
                      7    DEC1990    133.8    118.7
                      8    JAN1991    134.6    119.0
                      9    FEB1991    134.8    117.2
                     10    MAR1991    135.0    116.2
                     11    APR1991    135.2    116.0
                     12    MAY1991    135.6    116.5
                     13    JUN1991    136.0    116.3
                     14    JUL1991    136.2    116.0
```

Figure 2.3. Time Series Data Set Containing Two Series

Standard Form of a Time Series Data Set

The simple way the CPI and PPI time series are stored in the USPRICE data set in the preceding example is termed the *standard form* of a time series data set. A time series data set in standard form has the following characteristics:

- The data set contains one variable for each time series.
- The data set contains exactly one observation for each time period.
- The data set contains an ID variable or variables that identify the time period of each observation.
- The data set is sorted by the ID variables associated with date time values, so the observations are in time sequence.
- The data are equally spaced in time. That is, successive observations are a fixed time interval apart, so the data set can be described by a single sampling interval such as hourly, daily, monthly, quarterly, yearly, and so forth. This means that time series with different sampling frequencies are not mixed in the same SAS data set.

Most SAS/ETS procedures that process time series expect the input data set to contain time series in this standard form, and this is the simplest way to store time series in SAS data sets. There are more complex ways to represent time series in SAS data sets.

You can incorporate cross-sectional dimensions with BY groups, so that each BY group is like a standard form time series data set. This method is discussed in the section "Cross-sectional Dimensions and BY Groups."

You can interleave time series, with several observations for each time period identified by another ID variable. Interleaved time series data sets are used to store several series in the same SAS variable. Interleaved time series data sets are often used to store series of actual values, predicted values, and residuals, or series of forecast values and confidence limits for the forecasts. This is discussed in the section "Interleaved Time Series and the _TYPE_ Variable" later in this chapter.

Several Series with Different Ranges

Different time series can have values recorded over different time ranges. Since a SAS data set must have the same observations for all variables, when time series with different ranges are stored in the same data set, missing values must be used for the periods in which a series is not available.

Suppose that in the previous example you did not record values for CPI before August 1990 and did not record values for PPI after June 1991. The USPRICE data set could be read with the following statements:

```
data usprice;
   input date monyy7. cpi ppi;
   format date monyy7.;
datalines;
jun1990     . 114.3
jul1990     . 114.5
aug1990 131.6 116.5
sep1990 132.7 118.4
oct1990 133.5 120.8
nov1990 133.8 120.1
dec1990 133.8 118.7
jan1991 134.6 119.0
feb1991 134.8 117.2
mar1991 135.0 116.2
apr1991 135.2 116.0
may1991 135.6 116.5
jun1991 136.0 116.3
jul1991 136.2     .
;
```

The decimal points with no digits in the data records represent missing data and are read by the SAS System as missing value codes.

In this example, the time range of the USPRICE data set is June 1990 through July 1991, but the time range of the CPI variable is August 1990 through July 1991, and the time range of the PPI variable is June 1990 through June 1991.

SAS/ETS procedures ignore missing values at the beginning or end of a series. That is, the series is considered to begin with the first nonmissing value and end with the last nonmissing value.

Missing Values and Omitted Observations

Missing data can also occur within a series. Missing values that appear after the beginning of a time series and before the end of the time series are called *embedded missing values*.

Suppose that in the preceding example you did not record values for CPI for November 1990 and did not record values for PPI for both November 1990 and March 1991. The USPRICE data set could be read with the following statements.

```
data usprice;
   input date monyy. cpi ppi;
   format date monyy.;
datalines;
jun1990     . 114.3
jul1990     . 114.5
aug1990 131.6 116.5
sep1990 132.7 118.4
oct1990 133.5 120.8
nov1990     .     .
dec1990 133.8 118.7
jan1991 134.6 119.0
feb1991 134.8 117.2
mar1991 135.0     .
apr1991 135.2 116.0
may1991 135.6 116.5
jun1991 136.0 116.3
jul1991 136.2     .
;
```

In this example, the series CPI has one embedded missing value, and the series PPI has two embedded missing values. The ranges of the two series are the same as before.

Note that the observation for November 1990 has missing values for both CPI and PPI; there is no data for this period. This is an example of a *missing observation*.

You might ask why the data record for this period is included in the example at all, since the data record contains no data. However, if the data record for November 1990 were deleted from the example, this would cause an *omitted observation* in the USPRICE data set. SAS/ETS procedures expect input data sets to contain observations for a contiguous time sequence. If you omit observations from a time series data set and then try to analyze the data set with SAS/ETS procedures, the omitted observations will cause errors. When all data are missing for a period, a missing observation should be included in the data set to preserve the time sequence of the series.

Cross-sectional Dimensions and BY Groups

Often, a collection of time series are related by a cross-sectional dimension. For example, the national average U.S. consumer price index data shown in the previous example can be disaggregated to show price indexes for major cities. In this case there are several related time series: CPI for New York, CPI for Chicago, CPI for Los Angeles, and so forth. When these time series are considered one data set, the city whose price level is measured is a cross-sectional dimension of the data.

There are two basic ways to store such related time series in a SAS data set. The first way is to use a standard form time series data set with a different variable for each series.

For example, the following statements read CPI series for three major U.S. cities:

```
data citycpi;
   input date monyy7. cpiny cpichi cpila;
   format date monyy7.;
datalines;
nov1989  133.200  126.700  130.000
dec1989  133.300  126.500  130.600
jan1990  135.100  128.100  132.100
feb1990  135.300  129.200  133.600
mar1990  136.600  129.500  134.500
apr1990  137.300  130.400  134.200
may1990  137.200  130.400  134.600
jun1990  137.100  131.700  135.000
jul1990  138.400  132.000  135.600
;
```

The second way is to store the data in a time series cross-sectional form. In this form, the series for all cross sections are stored in one variable and a cross-section ID variable is used to identify observations for the different series. The observations are sorted by the cross-section ID variable and by time within each cross section.

The following statements indicate how to read the CPI series for U.S. cities in time series cross-sectional form:

```
data cpicity;
   input city $11. date monyy7. cpi;
   format date monyy7.;
datalines;
Chicago     nov1989  126.700
Chicago     dec1989  126.500
Chicago     jan1990  128.100
Chicago     feb1990  129.200
Chicago     mar1990  129.500
Chicago     apr1990  130.400
Chicago     may1990  130.400
Chicago     jun1990  131.700
Chicago     jul1990  132.000
Los Angeles nov1989  130.000
Los Angeles  dec1989  130.600
Los Angeles  jan1990  132.100
 ... etc. ...
New York     may1990  137.200
New York     jun1990  137.100
New York     jul1990  138.400
;

proc sort data=cpicity;
   by city date;
run;
```

When processing a time series cross-section-form data set with most SAS/ETS procedures, use the cross-section ID variable in a BY statement to process the time series separately. The data set must be sorted by the cross-section ID variable and sorted by date within each cross section. The PROC SORT step in the preceding example ensures that the CPICITY data set is correctly sorted.

When the cross-section ID variable is used in a BY statement, each BY group in the data set is like a standard form time series data set. Thus, SAS/ETS procedures that expect a standard form time series data set can process time series cross-sectional data sets when a BY statement is used, producing an independent analysis for each cross section.

It is also possible to analyze time series cross-sectional data jointly. The TSCSREG procedure expects the input data to be in the time series cross-sectional form described here. See Chapter 27 for more information.

Interleaved Time Series

Normally, a time series data set has only one observation for each time period, or one observation for each time period within a cross section for a time series cross-sectional form data set. However, it is sometimes useful to store several related time series in the same variable when the different series do not correspond to levels of a cross-sectional dimension of the data.

In this case, the different time series can be interleaved. An interleaved time series data set is similar to a time series cross-sectional data set, except that the observations are sorted differently, and the ID variable that distinguishes the different time series does not represent a cross-sectional dimension.

Some SAS/ETS procedures produce interleaved output data sets. The interleaved time series form is a convenient way to store procedure output when the results consist of several different kinds of series for each of several input series. (Interleaved time series are also easy to process with plotting procedures. See the section "Plotting Time Series" later in this chapter.)

For example, the FORECAST procedure fits a model to each input time series and computes predicted values and residuals from the model. The FORECAST procedure then uses the model to compute forecast values beyond the range of the input data and also to compute upper and lower confidence limits for the forecast values.

Thus, the output from PROC FORECAST consists of five related time series for each variable forecast. The five resulting time series for each input series are stored in a single output variable with the same name as the input series being forecast. The observations for the five resulting series are identified by values of the ID variable _TYPE_. These observations are interleaved in the output data set with observations for the same date grouped together.

The following statements show the use of PROC FORECAST to forecast the variable CPI in the USCPI data set. Figure 2.4 shows part of the output data set produced by PROC FORECAST and illustrates the interleaved structure of this data set.

```
proc forecast data=uscpi interval=month lead=12
              out=foreout outfull outresid;
   var cpi;
   id date;
run;

proc print data=foreout;
run;
```

```
Obs       date     _TYPE_      _LEAD_         cpi

 37    JUN1991    ACTUAL         0       136.000
 38    JUN1991    FORECAST       0       136.146
 39    JUN1991    RESIDUAL       0        -0.146
 40    JUL1991    ACTUAL         0       136.200
 41    JUL1991    FORECAST       0       136.566
 42    JUL1991    RESIDUAL       0        -0.366
 43    AUG1991    FORECAST       1       136.856
 44    AUG1991    L95            1       135.723
 45    AUG1991    U95            1       137.990
 46    SEP1991    FORECAST       2       137.443
 47    SEP1991    L95            2       136.126
 48    SEP1991    U95            2       138.761
```

Figure 2.4. Partial Listing of Output Data Set Produced by PROC FORECAST

Observations with _TYPE_=ACTUAL contain the values of CPI read from the input data set. Observations with _TYPE_=FORECAST contain one-step-ahead predicted values for observations with dates in the range of the input series, and contain forecast values for observations for dates beyond the range of the input series. Observations with _TYPE_=RESIDUAL contain the difference between the actual and one-step-ahead predicted values. Observations with _TYPE_=U95 and _TYPE_=L95 contain the upper and lower bounds of the 95% confidence interval for the forecasts.

Using Interleaved Data Sets as Input to SAS/ETS Procedures

Interleaved time series data sets are not directly accepted as input by SAS/ETS procedures. However, it is easy to use a WHERE statement with any procedure to subset the input data and select one of the interleaved time series as the input.

For example, to analyze the residual series contained in the PROC FORECAST output data set with another SAS/ETS procedure, include a WHERE

TYPE='RESIDUAL'; statement. The following statements perform a spectral analysis of the residuals produced by PROC FORECAST in the preceding example:

```
proc spectra data=foreout out=spectout;
   var cpi;
   where _type_='RESIDUAL';
run;
```

Combined Cross Sections and Interleaved Time Series Data Sets

Interleaved time series output data sets produced from BY-group processing of time series cross-sectional input data sets have a complex structure combining a cross-sectional dimension, a time dimension, and the values of the _TYPE_ variable. For example, consider the PROC FORECAST output data set produced by the following.

```
data cpicity;
   input city $11. date monyy7. cpi;
   format date monyy7.;
datalines;
Chicago      nov1989  126.700
Chicago      dec1989  126.500
Chicago      jan1990  128.100
 ... etc. ...
New York     may1990  137.200
New York     jun1990  137.100
New York     jul1990  138.400
;

proc sort data=cpicity;
   by city date;
run;

proc forecast data=cpicity interval=month lead=2
              out=foreout outfull outresid;
   var cpi;
   id date;
   by city;
run;
```

The output data set FOREOUT contains many different time series in the single variable CPI. BY groups identified by the variable CITY contain the result series for the different cities. Within each value of CITY, the actual, forecast, residual, and confidence limits series are stored in interleaved form, with the observations for the different series identified by the values of _TYPE_.

Output Data Sets of SAS/ETS Procedures

Some SAS/ETS procedures produce interleaved output data sets (like PROC FORECAST), while other SAS/ETS procedures produce standard form time series data sets. The form a procedure uses depends on whether the procedure is normally used to produce multiple result series for each of many input series in one step (as PROC FORECAST does).

The way different SAS/ETS procedures store result series in output data sets is summarized in Table 2.3.

Table 2.3. Form of Output Data Set for SAS/ETS Procedures

Procedures producing standard form output data sets with fixed names for result series: • ARIMA • SPECTRA • STATESPACE
Procedures producing standard form output data sets with result series named by an OUTPUT statement: • AUTOREG • PDLREG • SIMLIN • SYSLIN • X11
Procedures producing interleaved form output data sets: • FORECAST • MODEL

See the chapters for these procedures for details on the output data sets they create.

For example, the ARIMA procedure can output actual series, forecast series, residual series, and confidence limit series just as the FORECAST procedure does. The PROC ARIMA output data set uses the standard form because PROC ARIMA is designed for the detailed analysis of one series at a time and so only forecasts one series at a time.

The following statements show the use of the ARIMA procedure to produce a forecast of the USCPI data set. Figure 2.5 shows part of the output data set produced by the ARIMA procedure's FORECAST statement. (The printed output from PROC ARIMA is not shown.) Compare the PROC ARIMA output data set shown in Figure 2.5 with the PROC FORECAST output data set shown in Figure 2.4.

```
proc arima data=uscpi;
   identify var=cpi(1);
   estimate q=1;
   forecast id=date interval=month lead=12 out=arimaout;
run;
```

```
proc print data=arimaout;
run;
```

```
Obs       date     cpi     FORECAST      STD        L95        U95      RESIDUAL

 13     JUN1991   136.0    136.078    0.36160    135.369    136.787    -0.07816
 14     JUL1991   136.2    136.437    0.36160    135.729    137.146    -0.23725
 15     AUG1991     .      136.574    0.36160    135.865    137.283      .
 16     SEP1991     .      137.042    0.62138    135.824    138.260      .
```

Figure 2.5. Partial Listing of Output Data Set Produced by PROC ARIMA

The output data set produced by the ARIMA procedure's FORECAST statement stores the actual values in a variable with the same name as the input series, stores the forecast series in a variable named FORECAST, stores the residuals in a variable named RESIDUAL, stores the 95% confidence limits in variables named L95 and U95, and stores the standard error of the forecast in the variable STD.

This method of storing several different result series as a standard form time series data set is simple and convenient. However, it only works well for a single input series. The forecast of a single series can be stored in the variable FORECAST, but if two series are forecast, two different FORECAST variables are needed.

The STATESPACE procedure handles this problem by generating forecast variable names FOR1, FOR2, and so forth. The SPECTRA procedure uses a similar method. Names like FOR1, FOR2, RES1, RES2, and so forth require you to remember the order in which the input series are listed. This is why PROC FORECAST, which is designed to forecast a whole list of input series at once, stores its results in interleaved form.

Other SAS/ETS procedures are often used for a single input series but can also be used to process several series in a single step. Thus, they are not clearly like PROC FORECAST nor clearly like PROC ARIMA in the number of input series they are designed to work with. These procedures use a third method for storing multiple result series in an output data set. These procedures store output time series in standard form (like PROC ARIMA does) but require an OUTPUT statement to give names to the result series.

Time Series Periodicity and Time Intervals

A fundamental characteristic of time series data is how frequently the observations are spaced in time. How often the observations of a time series occur is called the *sampling frequency* or the *periodicity* of the series. For example, a time series with one observation each month has a monthly sampling frequency or monthly periodicity and so is called a monthly time series.

In the SAS System, data periodicity is described by specifying periodic *time intervals* into which the dates of the observations fall. For example, the SAS time interval MONTH divides time into calendar months.

Several SAS/ETS procedures enable you to specify the periodicity of the input data set with the INTERVAL= option. For example, specifying INTERVAL=MONTH indicates that the procedure should expect the ID variable to contain SAS date values, and that the date value for each observation should fall in a separate calendar month. The EXPAND procedure uses interval name values with the FROM= and TO= options to control the interpolation of time series from one periodicity to another.

The SAS System also uses time intervals in several other ways. In addition to indicating the periodicity of time series data sets, time intervals are used with the interval functions INTNX and INTCK, and for controlling the plot axis and reference lines for plots of data over time.

Specifying Time Intervals

Time intervals are specified in SAS Software using *interval names* like YEAR, QTR, MONTH, DAY, and so forth. Table 2.4 summarizes the basic types of intervals.

Table 2.4. Basic Interval Types

Name	**Periodicity**
YEAR	Yearly
SEMIYEAR	Semiannual
QTR	Quarterly
MONTH	Monthly
SEMIMONTH	1st and 16th of each month
TENDAY	1st, 11th, and 21st of each month
WEEK	Weekly
WEEKDAY	Daily ignoring weekend days
DAY	Daily
HOUR	Hourly
MINUTE	Every Minute
SECOND	Every Second

Interval names can be abbreviated in various ways. For example, you could specify monthly intervals as MONTH, MONTHS, MONTHLY, or just MON. The SAS System accepts all these forms as equivalent.

Interval names can also be qualified with a multiplier to indicate multiperiod intervals. For example, biennial intervals are specified as YEAR2.

Interval names can also be qualified with a shift index to indicate intervals with different starting points. For example, fiscal years starting in July are specified as YEAR.7.

Time intervals are classified as either date intervals or datetime intervals. Date intervals are used with SAS date values, while datetime intervals are used with SAS datetime values. The interval types YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, and DAY are date intervals. HOUR, MINUTE, and SECOND are datetime intervals. Date intervals can be turned into datetime intervals for use with datetime values by prefixing the interval name with 'DT'. Thus DTMONTH intervals are like MONTH intervals but are used with datetime ID values instead of date ID values.

See Chapter 3 for more information about specifying time intervals and for a detailed reference to the different kinds of intervals available.

Using Time Intervals with SAS/ETS Procedures

The ARIMA, FORECAST, and STATESPACE procedures use time intervals with the INTERVAL= option to specify the periodicity of the input data set. The EXPAND procedure uses time intervals with the FROM= and TO= options to specify the periodicity of the input and the output data sets. The DATASOURCE and CITIBASE procedures use the INTERVAL= option to control the periodicity of time series extracted from time series databases.

The INTERVAL= option (FROM= option for PROC EXPAND) is used with the ID statement to fully describe the observations that make up the time series. SAS/ETS procedures use the time interval specified by the INTERVAL= option and the ID variable in the following ways:

- to validate the data periodicity. The ID variable is used to check the data and verify that successive observations have valid ID values corresponding to successive time intervals.
- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for January 1990 is followed by an observation for April 1990, there is a gap in the input data with two omitted observations.
- to label forecast observations in the output data set. The values of the ID variable for the forecast observations after the end of the input data set are extrapolated according to the frequency specifications of the INTERVAL= option.

Time Intervals, the Time Series Forecasting System and the Time Series Viewer

Time intervals are used in the Time Series Forecasting System and Time Series Viewer to identify the number of seasonal cycles or seasonality associated with a DATE, DATETIME or TIME ID variable. For example, monthly time series have a seasonality of 12 because there are 12 months in a year; quarterly time series have a seasonality of 4 because there are 4 quarters in a year. The seasonality is used to analyze seasonal properties of time series data and to estimate seasonal forecasting methods.

Plotting Time Series

This section discusses SAS procedures available for plotting time series data. This section assumes you are generally familiar with SAS plotting procedures and only discusses certain aspects of the use of these procedures with time series data.

The Time Series Viewers displays and analyzes time series plots for time series data sets which do not contain cross-sections. Refer to the Chapter 34, "Getting Started with Time Series Forecasting," later in this book.

The GPLOT procedure produces high resolution color graphics plots. Refer to *SAS/GRAPH Software: Reference, Volume 1 and Volume 2* for information about the GPLOT procedure, SYMBOL statements, and other SAS/GRAPH features.

The PLOT procedure and the TIMEPLOT procedure produce low resolution line printer type plots. Refer to the *SAS Procedures Guide* for information about these procedures.

Using the Time Series Viewer

The following command starts the Time Series Viewer to display the plot of CPI in the USCPI data set against DATE. (The USCPI data set was shown in the previous example; the time series used in the following example contains more observations than previously shown.)

```
tsview data=uscpi var=cpi timeid=date
```

The TSVIEW DATA=option specifies the data set to be viewed; the VAR=option specifies the variable which contains the time series observations; the TIMEID=option specifies the time series ID variable.

Using PROC GPLOT

The following statements use the GPLOT procedure to plot CPI in the USCPI data set against DATE. (The USCPI data set was shown in a previous example; the data set plotted in the following example contains more observations than shown previously.) The SYMBOL statement is used to draw a smooth line between the plotted points and to specify the plotting character.

```
proc gplot data=uscpi;
   symbol i=spline v=circle h=2;
   plot cpi * date;
run;
```

The plot is shown in Figure 2.6.

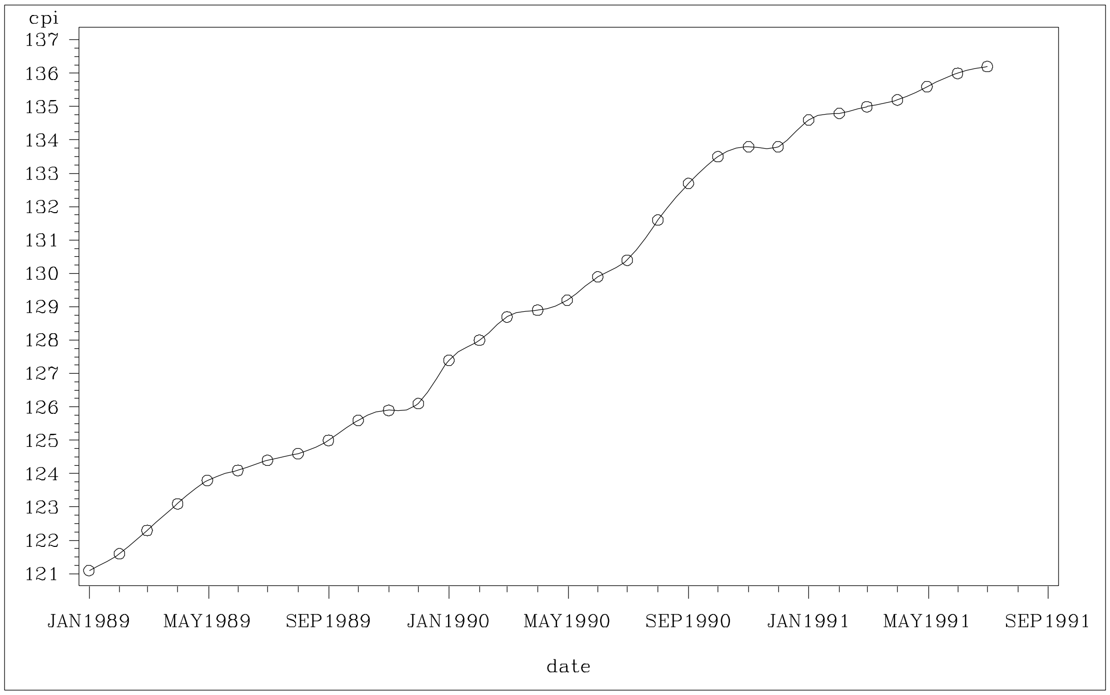


Figure 2.6. Plot of Monthly CPI Over Time

Controlling the Time Axis: Tick Marks and Reference Lines

It is possible to control the spacing of the tick marks on the time axis. The following statements use the HAXIS= option to tell PROC GPLOT to mark the axis at the start of each quarter. (The GPLOT procedure prints a warning message indicating that the intervals on the axis are not evenly spaced. This message simply reflects the fact that there is a different number of days in each quarter. This warning message can be ignored.)

```
proc gplot data=uscpi;
   symbol i=spline v=circle h=2;
   format date yyqc.;
   plot cpi * date /
        haxis= '1jan89'd to '1jul91'd by qtr;
run;
```

The plot is shown in Figure 2.7.

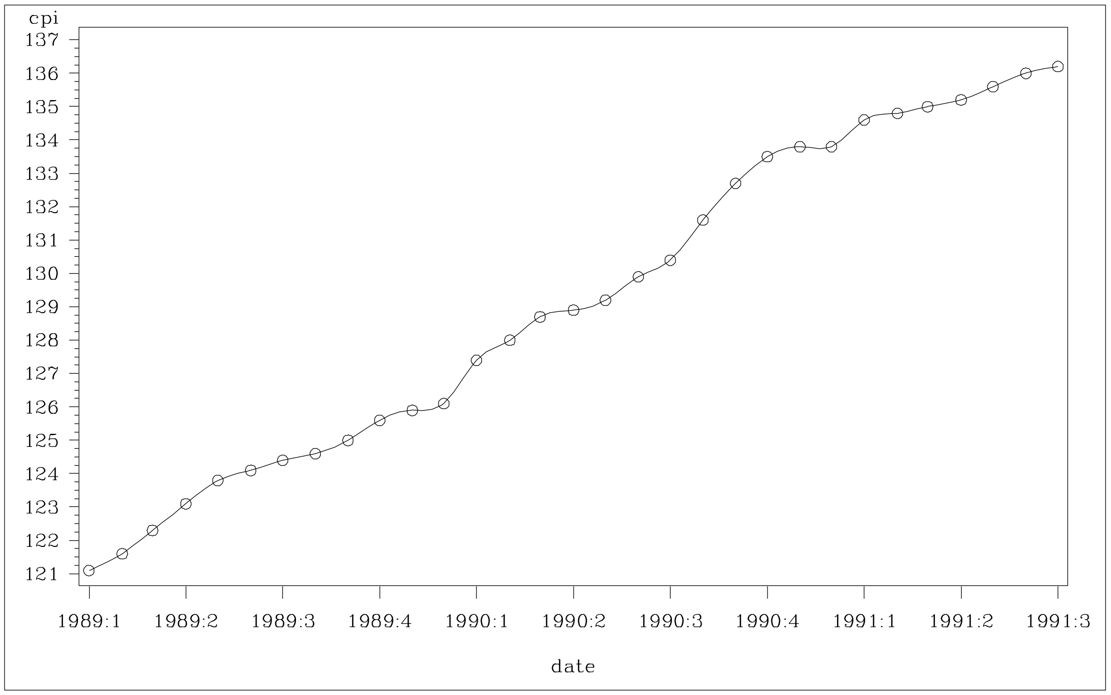


Figure 2.7. Plot of Monthly CPI Over Time

The following example changes the plot by using year and quarter value to label the tick marks. The FORMAT statement causes PROC GPLOT to use the YYQC format to print the date values. This example also shows how to place reference lines on the plot with the HREF= option. Reference lines are drawn to mark the boundary between years.

```
proc gplot data=uscpi;
   symbol i=spline v=circle h=2;
   plot cpi * date /
        haxis= '1jan89'd to '1jul91'd by qtr
        href= '1jan90'd to '1jan91'd by year;
   format date yyqc6.;
run;
```

The plot is shown in Figure 2.8.

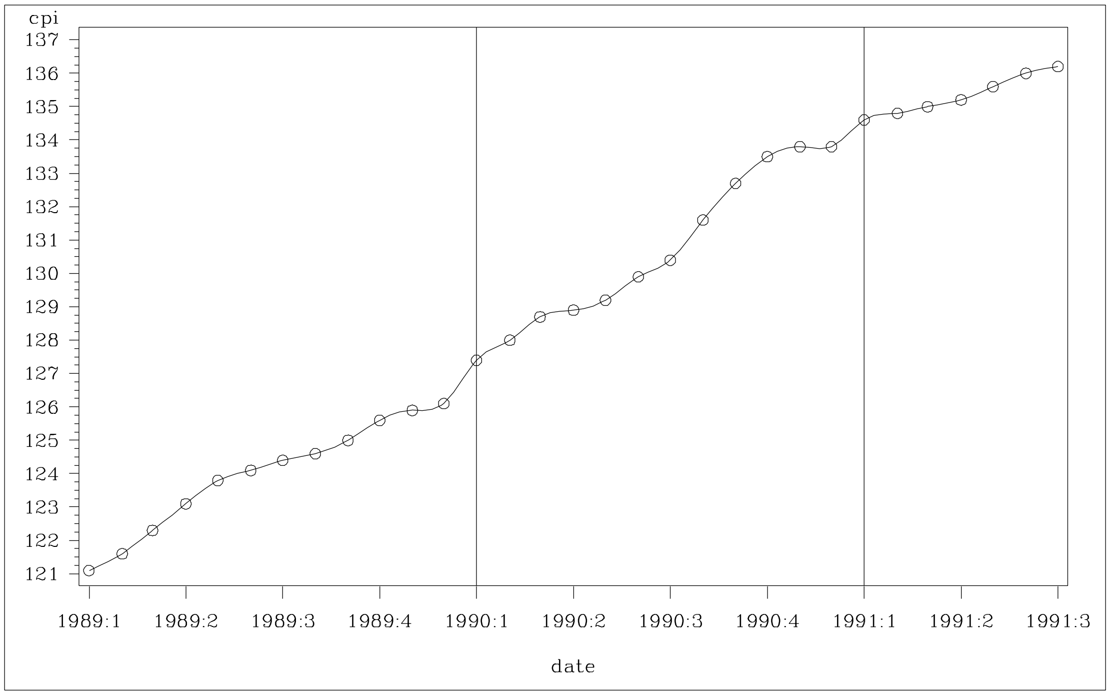


Figure 2.8. Plot of Monthly CPI Over Time

Overlay Plots of Different Variables

You can plot two or more series on the same graph. Plot series stored in different variables by specifying multiple plot requests on one PLOT statement, and use the OVERLAY option. Specify a different SYMBOL statement for each plot.

For example, the following statements plot the CPI, FORECAST, L95, and U95 variables produced by PROC ARIMA in a previous example. The SYMBOL1 statement is used for the actual series. Values of the actual series are labeled with a star, and the points are not connected. The SYMBOL2 statement is used for the forecast series. Values of the forecast series are labeled with an open circle, and the points are connected with a smooth curve. The SYMBOL3 statement is used for the upper and lower confidence limits series. Values of the upper and lower confidence limits points are not plotted, but a broken line is drawn between the points. A reference line is drawn to mark the start of the forecast period. Quarterly tick marks with YYQC format date values are used.

```
proc arima data=uscpi;
   identify var=cpi(1);
   estimate q=1;
   forecast id=date interval=month lead=12 out=arimaout;
run;

proc gplot data=arimaout;
   symbol1 i=none   v=star h=2;
   symbol2 i=spline v=circle h=2;
```

```
      symbol3 i=spline l=5;
      format date yyqc4.;
      plot cpi * date = 1
           forecast * date = 2
           ( l95 u95 ) * date = 3 /
           overlay
           haxis= '1jan89'd to '1jul92'd by qtr
           href= '15jul91'd ;
   run;
```

The plot is shown in Figure 2.9.

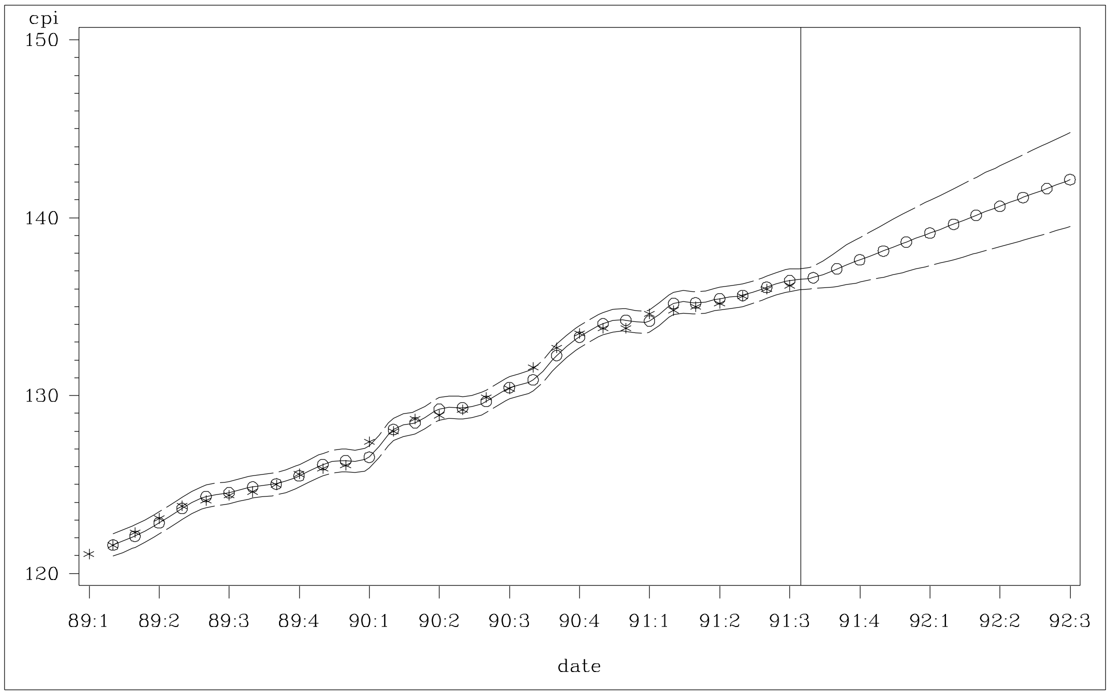


Figure 2.9. Plot of ARIMA Forecast

Overlay Plots of Interleaved Series

You can also plot several series on the same graph when the different series are stored in the same variable in interleaved form. Plot interleaved time series by using the values of the ID variable to distinguish the different series and by selecting different SYMBOL statements for each plot.

The following example plots the output data set produced by PROC FORECAST in a previous example. Since the residual series has a different scale than the other series, it is excluded from the plot with a WHERE statement.

The _TYPE_ variable is used on the PLOT statement to identify the different series and to select the SYMBOL statements to use for each plot. The first SYMBOL statement is used for the first sorted value of _TYPE_, which is _TYPE_=ACTUAL. The second SYMBOL statement is used for the second sorted value of the _TYPE_ variable (_TYPE_=FORECAST), and so forth.

```
proc forecast data=uscpi interval=month lead=12
              out=foreout outfull outresid;
   var cpi;
   id date;
run;

proc gplot data=foreout;
   symbol1 i=none   v=star h=2;
   symbol2 i=spline v=circle h=2;
   symbol3 i=spline l=20;
   symbol4 i=spline l=20;
   format date yyqc4.;
   plot cpi * date = _type_ /
        haxis= '1jan89'd to '1jul92'd by qtr
        href= '15jul91'd ;
   where _type_ ^= 'RESIDUAL';
run;
```

The plot is shown in Figure 2.10.

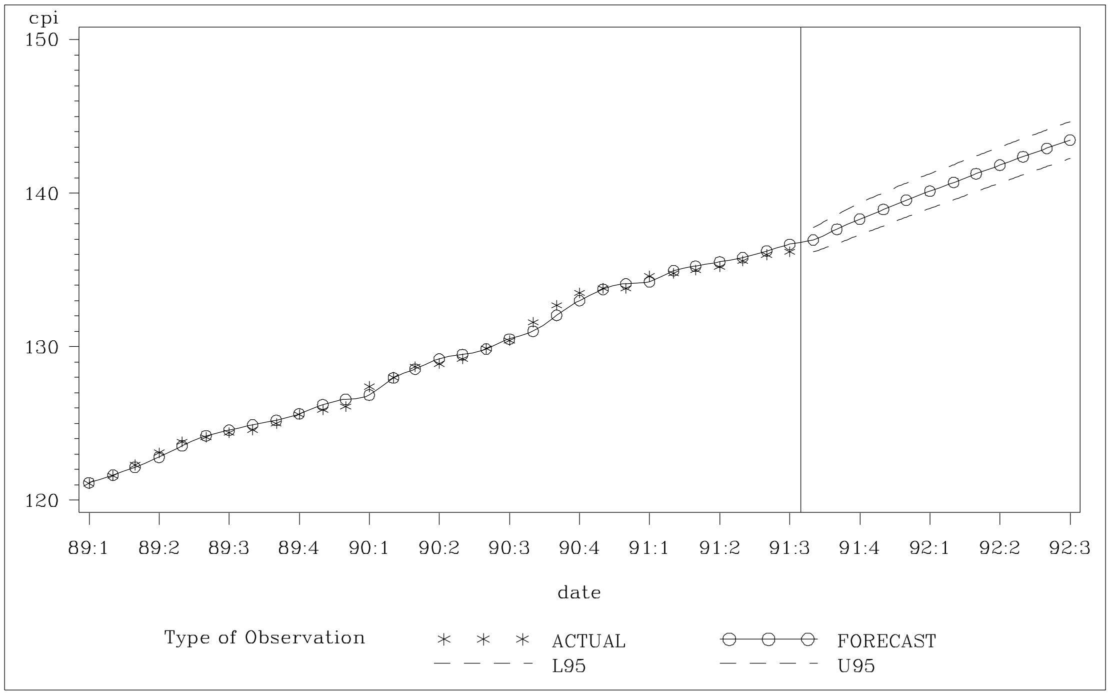


Figure 2.10. Plot of Forecast

Residual Plots

The following example plots the residuals series that was excluded from the plot in the previous example. The SYMBOL statement specifies a needle plot, so that each residual point is plotted as a vertical line showing deviation from zero.

```
proc gplot data=foreout;
   symbol1 i=needle v=circle width=6;
```

```
      format date yyqc4.;
      plot cpi * date /
           haxis= '1jan89'd to '1jul91'd by qtr ;
      where _type_ = 'RESIDUAL';
   run;
```

The plot is shown in Figure 2.11.

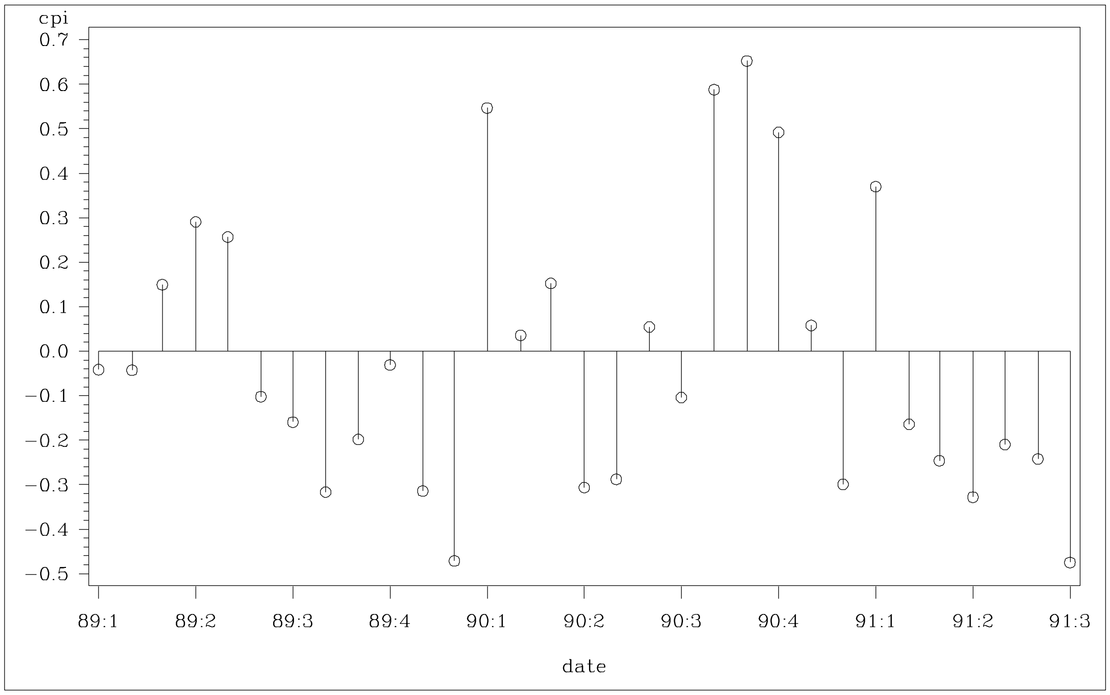


Figure 2.11. Plot of Residuals

Using PROC PLOT

The following statements use the PLOT procedure to plot CPI in the USCPI data set against DATE. (The data set plotted contains more observations than shown in the previous examples.) The plotting character used is a plus sign (+).

```
   proc plot data=uscpi;
      plot cpi * date = '+';
   run;
```

The plot is shown in Figure 2.12.

```
                     Plot of cpi*date.  Symbol used is '+'.

cpi |
140 +
    |
    |
    |
    |
    |                                                                ++ +
135 +                                                        + + + +
    |                                                     + +
    |                                                   +
    |                                                 +
    |                                               +
    |
130 +                                            ++
    |                                        + +
    |                                    + +
    |                                  +
    |
    |                             + + +
125 +                         + +
    |                    + ++
    |                  +
    |                +
    |              +
    |             +
120 +
    |
    --+-----------+-----------+-----------+-----------+-----------+-----------+-
   JUN1988     JAN1989     JUL1989     FEB1990     AUG1990     MAR1991   OCT1991

                                        date
```

Figure 2.12. Plot of Monthly CPI Over Time

Controlling the Time Axis: Tick Marks and Reference Lines

In the preceding example, the spacing of values on the time axis looks a bit odd in that the dates do not match for each year. Because DATE is a SAS date variable, the PLOT procedure needs additional instruction on how to place the time axis tick marks. The following statements use the HAXIS= option to tell PROC PLOT to mark the axis at the start of each quarter.

```
proc plot data=uscpi;
   plot cpi * date = '+' /
        haxis= '1jan89'd to '1jul91'd by qtr;
run;
```

The plot is shown in Figure 2.13.

```
                     Plot of cpi*date.  Symbol used is '+'.

140 +
    |
    |
    |
    |                                                                   + +  +
135 +                                                          + +  + +
    |                                                   +  + +
cpi |                                                 +
    |                                               +
    |
130 +                                          + +
    |                                   + +  +
    |                                 +
    |                              +
    |                       +  + +
125 +                  +  +
    |           +  + +
    |         +
    |    + +
    |  +
120 +
    ---+------+------+------+------+------+------+------+------+------+------+--
       J      A      J      O      J      A      J      O      J      A      J
       A      P      U      C      A      P      U      C      A      P      U
       N      R      L      T      N      R      L      T      N      R      L
       1      1      1      1      1      1      1      1      1      1      1
       9      9      9      9      9      9      9      9      9      9      9
       8      8      8      8      9      9      9      9      9      9      9
       9      9      9      9      0      0      0      0      1      1      1

                                        date
```

Figure 2.13. Plot of Monthly CPI Over Time

The following example improves the plot by placing tick marks every year and adds quarterly reference lines to the plot using the HREF= option. The FORMAT statement tells PROC PLOT to print just the year part of the date values on the axis. The plot is shown in Figure 2.14.

```
proc plot data=uscpi;
   plot cpi * date = '+' /
        haxis= '1jan89'd to '1jan92'd by year
        href=  '1apr89'd to '1apr91'd by qtr ;
   format date year4.;
run;
```

```
               Plot of cpi*date.  Symbol used is ’+’.

cpi |       |   |    |    |    |   |    |    |    |
140 +       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |+ ++
135 +       |   |    |    |    |   |    |    + ++ +
    |       |   |    |    |    |   |    | ++ |    |
    |       |   |    |    |    |   |    +    |    |
    |       |   |    |    |    |   |   +|    |    |
    |       |   |    |    |    |   | +  |    |    |
    |       |   |    |    |    |   |    |    |    |
130 +       |   |    |    |    |  ++    |    |    |
    |       |   |    |    |    ++  |    |    |    |
    |       |   |    |    | ++ |   |    |    |    |
    |       |   |    |    +    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    |       |   |    + ++ |    |   |    |    |    |
125 +       |   | + +|    |    |   |    |    |    |
    |       |+ ++    |    |    |   |    |    |    |
    |       +   |    |    |    |   |    |    |    |
    |     + |   |    |    |    |   |    |    |    |
    |    +  |   |    |    |    |   |    |    |    |
    |  +    |   |    |    |    |   |    |    |    |
120 +       |   |    |    |    |   |    |    |    |
    |       |   |    |    |    |   |    |    |    |
    ---+------------------+------------------+------------------+--
     1989               1990               1991               1992

                                  date
```

Figure 2.14. Plot of Monthly CPI Over Time

Marking the Subperiod of Points

In the preceding example, it is a little hard to tell which month each point is, although the quarterly reference lines help some. The following example shows how to set the plotting symbol to the first letter of the month name. A DATA step first makes a copy of DATE and gives this variable PCHAR a MONNAME1. format. The variable PCHAR is used in the PLOT statement to supply the plotting character.

This example also changes the plot by using quarterly tick marks and by using the YYQC format to print the date values. This example also changes the HREF= option to use annual reference lines. The plot is shown in Figure 2.15.

```
data temp;
   set uscpi;
   pchar = date;
   format pchar monname1.;
run;

proc plot data=temp;
   plot cpi * date = pchar /
        haxis= ’1jan89’d to ’1jul91’d by qtr
```

```
         href= '1jan90'd to '1jan91'd by year;
   format date yyqc4.;
run;
```

```
             Plot of cpi*date.  Symbol is value of pchar.

cpi |                          |                       |
140 +                          |                       |
    |                          |                       |
    |                          |                       |
    |                          |                       |
    |                          |                       |
    |                          |                       |       M J J
135 +                          |                       J F M A
    |                          |                   N D |
    |                          |                 O     |
    |                          |               S       |
    |                          |             A         |
    |                          |                       |
130 +                          |         J J           |
    |                          |     A M               |
    |                          | F M                   |
    |                          J                       |
    |                          |                       |
    |                    O N D |                       |
125 +                A S       |                       |
    |          M J J           |                       |
    |        A                 |                       |
    |      M                   |                       |
    |    F                     |                       |
    |  J                       |                       |
120 +                          |                       |
    |                          |                       |
    ---+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+--
     89:1  89:2  89:3  89:4  90:1  90:2  90:3  90:4  91:1  91:2  91:3

                                   date
```

Figure 2.15. Plot of Monthly CPI Over Time

Overlay Plots of Different Variables

Plot different series in different variables by specifying the different plot requests, each with its own plotting character, on the same PLOT statement, and use the OVERLAY option.

For example, the following statements plot the CPI, FORECAST, L95, and U95 variables produced by PROC ARIMA in a previous example. The actual series CPI is labeled with the plot character plus (+). The forecast series is labeled with the plot character F. The upper and lower confidence limits are labeled with the plot character period (.). The plot is shown in Figure 2.16.

```
proc arima data=uscpi;
   identify var=cpi(1);
   estimate q=1;
   forecast id=date interval=month lead=12 out=arimaout;
run;
```

```
proc plot data=arimaout;
   plot cpi * date = '+' forecast * date = 'F'
        ( l95 u95 ) * date = '.' /
        overlay
        haxis= '1jan89'd to '1jul92'd by qtr
        href= '1jan90'd to '1jan92'd by year ;
run;
```

```
                  Plot of cpi*date.       Symbol used is '+'.
                  Plot of FORECAST*date.  Symbol used is 'F'.
                  Plot of L95*date.       Symbol used is '.'.
                  Plot of U95*date.       Symbol used is '.'.

cpi |                      |                   |                   |
150 +                      |                   |                   |
    |                      |                   |                   |
    |                      |                   |                   |
    |                      |                   |                   |     . . .
    |                      |                   |                   | .. .F F F
140 +                      |                   |                .. F FF F  . .
    |                      |                   |         . .F F FF . .. ..
    |                      |                   | F ++ + ++ F. . .. |
    |                      |              + + ++ + ..              |
    |                      |         . + +.    |                   |
130 +                      |   .F + ++ F       |                   |
    |                      + + ++ .            |                   |
    |            . .+ + + +F                   |                   |
    |       ++ + + +. .    |                   |                   |
    |  ++ + F              |                   |                   |
120 +   .                  |                   |                   |
    |                      |                   |                   |
    ---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+--
       J    A    J    O    J    A    J    O    J    A    J    O    J    A    J
       A    P    U    C    A    P    U    C    A    P    U    C    A    P    U
       N    R    L    T    N    R    L    T    N    R    L    T    N    R    L
       1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
       9    9    9    9    9    9    9    9    9    9    9    9    9    9    9
       8    8    8    8    9    9    9    9    9    9    9    9    9    9    9
       9    9    9    9    0    0    0    0    1    1    1    1    2    2    2

                                        date

NOTE: 15 obs had missing values.  77 obs hidden.
```

Figure 2.16. Plot of ARIMA Forecast

Overlay Plots of Interleaved Series

Plot interleaved time series by using the first character of the ID variable to distinguish the different series as the plot character.

The following example plots the output data set produced by PROC FORECAST in a previous example. The _TYPE_ variable is used on the PLOT statement to supply plotting characters to label the different series.

The actual series is plotted with A, the forecast series is plotted with F, the lower confidence limit is plotted with L, and the upper confidence limit is plotted with U.

Since the residual series has a different scale than the other series, it is excluded from the plot with a WHERE statement. The plot is shown in Figure 2.17.

```
proc forecast data=uscpi interval=month lead=12
              out=foreout outfull outresid;
   var cpi;
   id date;
run;

proc plot data=foreout;
   plot cpi * date = _type_ /
        haxis= '1jan89'd to '1jul92'd by qtr
        href= '1jan90'd to '1jan92'd by year ;
   where _type_ ^= 'RESIDUAL';
run;
```

```
                 Plot of cpi*date.  Symbol is value of _TYPE_.

cpi |                      |                   |                   |
150 +                      |                   |                   |
    |                      |                   |                   |
    |                      |                   |                   |
    |                      |                   |                   |         U
    |                      |                   |                   |    UU F F
    |                      |                   |                   U UF FF L L
140 +                      |                   |              U UF F FL L
    |                      |                   |           UF F FL L
    |                      |                   |        AA FL L    |
    |                      |                   A A AA A            |
    |                      |             AA A A|                   |
    |                      |           A F     |                   |
130 +                      |    F A AA         |                   |
    |                      | A AA              |                   |
    |                   A AA                   |                   |
    |          F A AA A    |                   |                   |
    |       AA A           |                   |                   |
    |  AA A                |                   |                   |
120 +                      |                   |                   |
    |                      |                   |                   |
    ---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+--
       J    A    J    O    J    A    J    O    J    A    J    O    J    A    J
       A    P    U    C    A    P    U    C    A    P    U    C    A    P    U
       N    R    L    T    N    R    L    T    N    R    L    T    N    R    L
       1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
       9    9    9    9    9    9    9    9    9    9    9    9    9    9    9
       8    8    8    8    9    9    9    9    9    9    9    9    9    9    9
       9    9    9    9    0    0    0    0    1    1    1    1    2    2    2

                                        date

NOTE: 36 obs hidden.
```

Figure 2.17. Plot of Forecast

Residual Plots

The following example plots the residual series that was excluded from the plot in the previous example. The VREF=0 option is used to draw a reference line at 0 on the vertical axis. The plot is shown in Figure 2.18.

```
proc plot data=foreout;
   plot cpi * date = ’*’ /
        vref=0
        haxis= ’1jan89’d to ’1jul91’d by qtr
        href= ’1jan90’d to ’1jan91’d by year ;
   where _type_ = ’RESIDUAL’;
run;
```

```
                     Plot of cpi*date.  Symbol used is ’*’.

 cpi |                             |                           |
 1.0 +                             |                           |
     |                             |                           |
     |                             |                           |
     |                             |                           |
     |                             |                  *        |
     |                             *                *          |
 0.5 +                             |                    *      |
     |                             |                           |
     |                             |                           *
     |        * *                  |                           |
     |     *                       |    *                      |
     |                             |           *           *   |
 0.0 +-*--------------------*------+--*------------------------+----------------
     |   *         *               |             *             |
     |               *    *        |                           | *
     |                             |         *                 |    *   * *
     |                 *       *   |      *                  * |      *
     |                             |                           |
-0.5 +                           * |                           |             *
     |                             |                           |
   --+------+------+------+------+------+------+------+------+------+------+--
     J      A      J      O      J      A      J      O      J      A      J
     A      P      U      C      A      P      U      C      A      P      U
     N      R      L      T      N      R      L      T      N      R      L
     1      1      1      1      1      1      1      1      1      1      1
     9      9      9      9      9      9      9      9      9      9      9
     8      8      8      8      9      9      9      9      9      9      9
     9      9      9      9      0      0      0      0      1      1      1

                                       date
```

Figure 2.18. Plot of Residuals

Using PROC TIMEPLOT

The TIMEPLOT procedure plots time series data vertically on the page instead of horizontally across the page as the PLOT procedure does. PROC TIMEPLOT can also print the data values as well as plot them.

The following statements use the TIMEPLOT procedure to plot CPI in the USCPI data set. Only the last 14 observations are included in this example. The plot is shown in Figure 2.19.

```
proc timeplot data=uscpi;
   plot cpi;
   id date;
   where date >= ’1jun90’d;
run;
```

```
   date        cpi      min                                                  max
                        129.9                                              136.2
                       *-------------------------------------------------------*
JUN1990     129.90     |c                                                      |
JUL1990     130.40     |    c                                                  |
AUG1990     131.60     |               c                                       |
SEP1990     132.70     |                        c                              |
OCT1990     133.50     |                               c                       |
NOV1990     133.80     |                                  c                    |
DEC1990     133.80     |                                  c                    |
JAN1991     134.60     |                                         c             |
FEB1991     134.80     |                                           c           |
MAR1991     135.00     |                                             c         |
APR1991     135.20     |                                              c        |
MAY1991     135.60     |                                                  c    |
JUN1991     136.00     |                                                     c |
JUL1991     136.20     |                                                      c|
                       *-------------------------------------------------------*
```

Figure 2.19. Output Produced by PROC TIMEPLOT

The TIMEPLOT procedure has several interesting features not discussed here. Refer to "The TIMEPLOT Procedure" in the *SAS Procedures Guide* for more information.

Calendar and Time Functions

Calendar and time functions convert calendar and time variables like YEAR, MONTH, DAY, and HOUR, MINUTE, SECOND into SAS date or datetime values, and vice versa.

The SAS calendar and time functions are DATEJUL, DATEPART, DAY, DHMS, HMS, HOUR, JULDATE, MDY, MINUTE, MONTH, QTR, SECOND, TIMEPART, WEEKDAY, YEAR, and YYQ. Refer to *SAS Language Reference* for more details about these functions.

Computing Dates from Calendar Variables

The MDY function converts MONTH, DAY, and YEAR values to a SAS date value. For example, MDY(10,17,91) returns the SAS date value ’17OCT91’D.

The YYQ function computes the SAS date for the first day of a quarter. For example, YYQ(91,4) returns the SAS date value ’1OCT91’D.

The DATEJUL function computes the SAS date for a Julian date. For example, DATEJUL(91290) returns the SAS date ’17OCT91’D.

The YYQ and MDY functions are useful for creating SAS date variables when the ID values recorded in the data are year and quarter; year and month; or year, month, and day, instead of dates that can be read with a date informat.

For example, the following statements read quarterly estimates of the gross national product of the U.S. from 1990:I to 1991:II from data records on which dates are coded as separate year and quarter values. The YYQ function is used to compute the variable DATE.

```
data usecon;
   input year qtr gnp;
   date = yyq( year, qtr );
   format date yyqc.;
datalines;
1990 1 5375.4
1990 2 5443.3
1990 3 5514.6
1990 4 5527.3
1991 1 5557.7
1991 2 5615.8
;
```

The monthly USCPI data shown in a previous example contained time ID values represented in the MONYY format. If the data records instead contain separate year and month values, the data can be read in and the DATE variable computed with the following statements:

```
data uscpi;
   input month year cpi;
   date = mdy( month, 1, year );
   format date monyy.;
datalines;
6 90 129.9
7 90 130.4
8 90 131.6
 ... etc. ...
;
```

Computing Calendar Variables from Dates

The functions YEAR, MONTH, DAY, WEEKDAY, and JULDATE compute calendar variables from SAS date values.

Returning to the example of reading the USCPI data from records containing date values represented in the MONYY format, you can find the month and year of each observation from the SAS dates of the observations using the following statements.

```
data uscpi;
   input date monyy7. cpi;
   format date monyy7.;
   year  = year( date );
   month = month( date );
datalines;
jun1990 129.9
jul1990 130.4
aug1990 131.6
sep1990 132.7
 ... etc. ...
;
```

Converting between Date, Datetime, and Time Values

The DATEPART function computes the SAS date value for the date part of a SAS datetime value. The TIMEPART function computes the SAS time value for the time part of a SAS datetime value.

The HMS function computes SAS time values from HOUR, MINUTE, and SECOND time variables. The DHMS function computes a SAS datetime value from a SAS date value and HOUR, MINUTE, and SECOND time variables.

See the "Date, Time, and Datetime Functions" section on page 127 for more information on the syntax of these functions.

Computing Datetime Values

To compute datetime ID values from calendar and time variables, first compute the date and then compute the datetime with DHMS.

For example, suppose you read tri-hourly temperature data with time recorded as YEAR, MONTH, DAY, and HOUR. The following statements show how to compute the ID variable DATETIME:

```
data weather;
   input year month day hour temp;
   datetime = dhms( mdy( month, day, year ), hour, 0, 0 );
   format datetime datetime10.;
datalines;
91 10 16 21  61
91 10 17  0  56
```

```
91 10 17  3  53
91 10 17  6  54
91 10 17  9  65
91 10 17 12  72
 ... etc. ...
;
```

Computing Calendar and Time Variables

The functions HOUR, MINUTE, and SECOND compute time variables from SAS datetime values. The DATEPART function and the date-to-calendar variables functions can be combined to compute calendar variables from datetime values.

For example, suppose the date and time of the tri-hourly temperature data in the preceding example were recorded as datetime values in the datetime format. The following statements show how to compute the YEAR, MONTH, DAY, and HOUR of each observation and include these variables in the SAS data set:

```
data weather;
   input datetime datetime13. temp;
   format datetime datetime10.;
   hour = hour( datetime );
   date = datepart( datetime );
   year = year( date );
   month = month( date );
   day = day( date );
datalines;
16oct91:21:00  61
17oct91:00:00  56
17oct91:03:00  53
17oct91:06:00  54
17oct91:09:00  65
17oct91:12:00  72
 ... etc. ...
;
```

Interval Functions INTNX and INTCK

The SAS interval functions INTNX and INTCK perform calculations with date, datetime values, and time intervals. They can be used for calendar calculations with SAS date values, to count time intervals between dates, and to increment dates or datetime values by intervals.

The INTNX function increments dates by intervals. INTNX computes the date or datetime of the start of the interval a specified number of intervals from the interval containing a given date or datetime value.

The form of the INTNX function is

INTNX(*interval, from, n* <, *alignment* >)

where:

interval	is a character constant or variable containing an interval name.
from	is a SAS date value (for date intervals) or datetime value (for datetime intervals).
n	is the number of intervals to increment from the interval containing the *from* value.
alignment	controls the alignment of SAS dates, within the interval, used to identify output observations. Can take the values BEGINNING|B, MIDDLE|M, or END|E.

The number of intervals to increment, *n*, can be positive, negative, or zero.

For example, the statement NEXTMON = INTNX('MONTH',DATE,1); assigns to the variable NEXTMON the date of the first day of the month following the month containing the value of DATE.

The INTCK function counts the number of interval boundaries between two dates or between two datetime values.

The form of the INTCK function is

INTCK(*interval*, *from*, *to*)

where:

interval	is a character constant or variable containing an interval name
from	is the starting date (for date intervals) or datetime value (for datetime intervals)
to	is the ending date (for date intervals) or datetime value (for datetime intervals).

For example, the statement NEWYEARS = INTCK('YEAR',DATE1,DATE2); assigns to the variable NEWYEARS the number of New Year's Days between the two dates.

Incrementing Dates by Intervals

Use the INTNX function to increment dates by intervals. For example, suppose you want to know the date of the start of the week that is six weeks from the week of 17 October 1991. The function INTNX('WEEK','17OCT91'D,6) returns the SAS date value '24NOV1991'D.

One practical use of the INTNX function is to generate periodic date values. For example, suppose the monthly U.S. Consumer Price Index data in a previous example were recorded without any time identifier on the data records. Given that you know the first observation is for June 1990, the following statements use the INTNX function to compute the ID variable DATE for each observation:

```
data uscpi;
   input cpi;
   date = intnx( 'month', '1jun1990'd, _n_-1 );
   format date monyy7.;
datalines;
129.9
130.4
131.6
132.7
 ... etc. ...
;
```

The automatic variable _N_ counts the number of times the DATA step program has executed, and in this case _N_ contains the observation number. Thus _N_-1 is the increment needed from the first observation date. Alternatively, we could increment from the month before the first observation, in which case the INTNX function in this example would be written INTNX('MONTH','1MAY1990'D,_N_).

Alignment of SAS Dates

Any date within the time interval corresponding to an observation of a periodic time series can serve as an ID value for the observation. For example, the USCPI data in a previous example might have been recorded with dates at the 15th of each month. The person recording the data might reason that since the CPI values are monthly averages, midpoints of the months might be the appropriate ID values.

However, as far as SAS/ETS procedures are concerned, what is important about monthly data is the month of each observation, not the exact date of the ID value. If you indicate that the data are monthly (with an INTERVAL=MONTH) option, SAS/ETS procedures ignore the day of the month in processing the ID variable. The MONYY format also ignores the day of the month.

Thus, you could read in the monthly USCPI data with midmonth DATE values using the following statements:

```
data uscpi;
   input date date9. cpi;
   format date monyy7.;
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6
15sep1990 132.7
 ... etc. ...
;
```

The results of using this version of the USCPI data set for analysis with SAS/ETS procedures would be the same as with first-of-month values for DATE. Although you can use any date within the interval as an ID value for the interval, you may find working with time series in SAS less confusing if you always use date ID values normalized to the start of the interval.

For some applications it may be preferable to use end of period dates, such as 31Jan1994, 28Feb1994, 31Mar1994, ..., 31Dec1994. For other applications, such as plotting time series, it may be more convenient to use interval midpoint dates to identify the observations.

SAS/ETS procedures provide an ALIGN= option to control the alignment of dates for output time series observations. Procedures supporting the ALIGN= option are ARIMA, DATASOURCE, EXPAND, and FORECAST. In addition, the INTNX library function supports an optional argument to specify the alignment of the returned date value.

To normalize date values to the start of intervals, use the INTNX function with a 0 increment. The INTNX function with an increment of 0 computes the date of the first day of the interval (or the first second of the interval for datetime values).

For example, INTNX('MONTH','17OCT1991'D,0, BEG) returns the date '1OCT1991'D'.

The following statements show how the preceding example can be changed to normalize the mid-month DATE values to first-of-month and end-of-month values. For exposition, the first-of-month value is transformed back into a middle-of-month value.

```
data uscpi;
   input date date9. cpi;
   format date monyy7.;
   monthbeg = intnx( 'month', date, 0, beg );
   midmonth = intnx( 'month', monthbeg, 0, mid );
   monthend = intnx( 'month', date, 0, end );
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6
15sep1990 132.7
 ... etc. ...
;
```

If you want to compute the date of a particular day within an interval, you can use calendar functions, or you can increment the starting date of the interval by a number of days. The following example shows three ways to compute the 7th day of the month:

```
data test;
   set uscpi;
   mon07_1 = mdy( month(date), 7, year(date) );
   mon07_2 = intnx( 'month', date, 0, beg ) + 6;
   mon07_3 = intnx( 'day', date, 6 );
run;
```

Computing the Width of a Time Interval

To compute the width of a time interval, subtract the ID value of the start of the next interval from the ID value of the start of the current interval. If the ID values are SAS dates, the width will be in days. If the ID values are SAS datetime values, the width will be in seconds.

For example, the following statements show how to add a variable WIDTH to the USCPI data set that contains the number of days in the month for each observation:

```
data uscpi;
   input date date9. cpi;
   format date monyy7.;
   width = intnx( 'month', date, 1 ) - intnx( 'month', date, 0 );
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6
15sep1990 132.7
 ... etc. ...
;
```

Computing the Ceiling of an Interval

To shift a date to the start of the next interval if not already at the start of an interval, subtract 1 from the date and use INTNX to increment the date by 1 interval.

For example, the following statements add the variable NEWYEAR to the monthly USCPI data set. The variable NEWYEAR contains the date of the next New Year's Day. NEWYEAR contains the same value as DATE when the DATE value is the start of year and otherwise contains the date of the start of the next year.

```
data test;
   set uscpi;
   newyear = intnx( 'year', date - 1, 1 );
   format newyear date.;
run;
```

Counting Time Intervals

Use the INTCK function to count the number of interval boundaries between two dates.

Note that the INTCK function counts the number of times the beginning of an interval is reached in moving from the first date to the second. It does not count the number of complete intervals between two dates.

For example, the function INTCK('MONTH','1JAN1991'D,'31JAN1991'D) returns 0, since the two dates are within the same month.

The function INTCK('MONTH','31JAN1991'D,'1FEB1991'D) returns 1, since the two dates lie in different months that are one month apart.

When the first date is later than the second date, INTCK returns a negative count. For example, the function INTCK('MONTH','1FEB1991'D,'31JAN1991'D) returns -1.

The following example shows how to use the INTCK function to count the number of Sundays, Mondays, Tuesdays, and so forth, in each month. The variables NSUNDAY, NMONDAY, NTUESDAY, and so forth, are added to the USCPI data set.

```
data uscpi;
   set uscpi;
   d0 = intnx( 'month', date, 0 ) - 1;
   d1 = intnx( 'month', date, 1 ) - 1;
   nsunday  = intck( 'week.1', d0, d1 );
   nmonday  = intck( 'week.2', d0, d1 );
   ntuesday = intck( 'week.3', d0, d1 );
   nwedday  = intck( 'week.4', d0, d1 );
   nthurday = intck( 'week.5', d0, d1 );
   nfriday  = intck( 'week.6', d0, d1 );
   nsatday  = intck( 'week.7', d0, d1 );
   drop d0 d1;
run;
```

Since the INTCK function counts the number of interval beginning dates between two dates, the number of Sundays is computed by counting the number of week boundaries between the last day of the previous month and the last day of the current month. To count Mondays, Tuesdays, and so forth, shifted week intervals are used. The interval type WEEK.2 specifies weekly intervals starting on Mondays, WEEK.3 specifies weeks starting on Tuesdays, and so forth.

Checking Data Periodicity

Suppose you have a time series data set, and you want to verify that the data periodicity is correct, the observations are dated correctly, and the data set is sorted by date. You can use the INTCK function to compare the date of the current observation with the date of the previous observation and verify that the dates fall into consecutive time intervals.

For example, the following statements verify that the data set USCPI is a correctly dated monthly data set. The RETAIN statement is used to hold the date of the previous observation, and the automatic variable _N_ is used to start the verification process with the second observation.

```
data _null_;
   set uscpi;
   retain prevdate;
   if _n_ > 1 then
      if intck( 'month', prevdate, date ) ^= 1 then
         put "Bad date sequence at observation number " _n_;
   prevdate = date;
run;
```

Filling in Omitted Observations in a Time Series Data Set

Recall that most SAS/ETS procedures expect input data to be in the standard form, with no omitted observations in the sequence of time periods. When data are missing for a time period, the data set should contain a missing observation, in which all variables except the ID variables have missing values.

You can replace omitted observations in a time series data set with missing observations by merging the data set with a data set containing a complete sequence of dates.

The following statements create a monthly data set, OMITTED, from data lines containing records for an intermittent sample of months. (Data values are not shown.) This data set is converted to a standard form time series data set in four steps.

First, the OMITTED data set is sorted to make sure it is in time order. Second, the first and last date in the data set are determined and stored in the data set RANGE. Third, the data set DATES is created containing only the variable DATE and containing monthly observations for the needed time span. Finally, the data sets OMITTED and DATES are merged to produce a standard form time series data set with missing observations inserted for the omitted records.

```
data omitted;
   input date monyy7. x y z;
   format date monyy7.;
datalines;
jan1991  ...
mar1991  ...
apr1991  ...
jun1991  ...
 ... etc. ...
;

proc sort data=omitted;
   by date;
run;


data range;
   retain from to;
   set omitted end=lastobs;
   if _n_ = 1 then from = date;
   if lastobs then do;
      to = date;
      output;
      end;
run;

data dates;
   set range;
   date = from;
   do while( date <= to );
      output;
```

```
         date = intnx( 'month', date, 1 );
         end;
      keep date;
   run;

   data standard;
      merge omitted dates;
      by date;
   run;
```

Using Interval Functions for Calendar Calculations

With a little thought, you can come up with a formula involving INTNX and INTCK functions and different interval types to perform almost any calendar calculation.

For example, suppose you want to know the date of the third Wednesday in the month of October 1991. The answer can be computed as

```
   intnx( 'week.4', '1oct91'd - 1, 3 )
```

which returns the SAS date value ’16OCT91’D.

Consider this more complex example: how many weekdays are there between 17 October 1991 and the second Friday in November 1991, inclusive? The following formula computes the number of weekdays between the date value contained in the variable DATE and the second Friday of the following month (including the ending dates of this period):

```
   n = intck( 'weekday', date - 1,
       intnx( 'week.6', intnx( 'month', date, 1 ) - 1, 2 ) + 1 );
```

Setting DATE to ’17OCT91’D and applying this formula produces the answer, N=17.

Lags, Leads, Differences, and Summations

When working with time series data, you sometimes need to refer to the values of a series in previous or future periods. For example, the usual interest in the consumer price index series shown in previous examples is how fast the index is changing, rather than the actual level of the index. To compute a percent change, you need both the current and the previous values of the series. When modeling a time series, you may want to use the previous values of other series as explanatory variables.

This section discusses how to use the DATA step to perform operations over time: lags, differences, leads, summations over time, and percent changes.

The EXPAND procedure can also be used to perform many of these operations; see Chapter 16, “The EXPAND Procedure,” for more information. See also the section "Transforming Time Series" later in this chapter.

The LAG and DIF Functions

The DATA step provides two functions, LAG and DIF, for accessing previous values of a variable or expression. These functions are useful for computing lags and differences of series.

For example, the following statements add the variables CPILAG and CPIDIF to the USCPI data set. The variable CPILAG contains lagged values of the CPI series. The variable CPIDIF contains the changes of the CPI series from the previous period; that is, CPIDIF is CPI minus CPILAG. The new data set is shown in part in Figure 2.20.

```
data uscpi;
   set uscpi;
   cpilag = lag( cpi );
   cpidif = dif( cpi );
run;

proc print data=uscpi;
run;
```

```
                  Obs    date      cpi     cpilag    cpidif

                    1    JUN90    129.9       .         .
                    2    JUL90    130.4     129.9      0.5
                    3    AUG90    131.6     130.4      1.2
                    4    SEP90    132.7     131.6      1.1
                    5    OCT90    133.5     132.7      0.8
                    6    NOV90    133.8     133.5      0.3
                    7    DEC90    133.8     133.8      0.0
                    8    JAN91    134.6     133.8      0.8
```

Figure 2.20. USCPI Data Set with Lagged and Differenced Series

Understanding the DATA Step LAG and DIF Functions

When used in this simple way, LAG and DIF act as lag and difference functions. However, it is important to keep in mind that, despite their names, the LAG and DIF functions available in the DATA step are not true lag and difference functions.

Rather, LAG and DIF are queuing functions that remember and return argument values from previous calls. The LAG function remembers the value you pass to it and returns as its result the value you passed to it on the previous call. The DIF function works the same way but returns the difference between the current argument and the remembered value. (LAG and DIF return a missing value the first time the function is called.)

A true lag function does not return the value of the argument for the "previous call," as do the DATA step LAG and DIF functions. Instead, a true lag function returns the value of its argument for the "previous observation," regardless of the sequence of previous calls to the function. Thus, for a true lag function to be possible, it must be clear what the "previous observation" is.

If the data are sorted chronologically, then LAG and DIF act as true lag and difference functions. If in doubt, use PROC SORT to sort your data prior to using the LAG and DIF functions. Beware of missing observations, which may cause LAG and DIF to return values that are not the actual lag and difference values

The DATA step is a powerful tool that can read any number of observations from any number of input files or data sets, can create any number of output data sets, and can write any number of output observations to any of the output data sets, all in the same program. Thus, in general, it is not clear what "previous observation" means in a DATA step program. In a DATA step program, the "previous observation" exists only if you write the program in a simple way that makes this concept meaningful.

Since, in general, the previous observation is not clearly defined, it is not possible to make true lag or difference functions for the DATA step. Instead, the DATA step provides queuing functions that make it easy to compute lags and differences.

Pitfalls of DATA Step LAG and DIF Functions

The LAG and DIF functions compute lags and differences provided that the sequence of calls to the function corresponds to the sequence of observations in the output data set. However, any complexity in the DATA step that breaks this correspondence causes the LAG and DIF functions to produce unexpected results.

For example, suppose you want to add the variable CPILAG to the USCPI data set, as in the previous example, and you also want to subset the series to 1991 and later years. You might use the following statements:

```
data subset;
   set uscpi;
   if date >= '1jan1991'd;
   cpilag = lag( cpi );  /* WRONG PLACEMENT! */
run;
```

If the subsetting IF statement comes before the LAG function call, the value of CPILAG will be missing for January 1991, even though a value for December 1990 is available in the USCPI data set. To avoid losing this value, you must rearrange the statements to ensure that the LAG function is actually executed for the December 1990 observation.

```
data subset;
   set uscpi;
   cpilag = lag( cpi );
   if date >= '1jan1991'd;
run;
```

In other cases, the subsetting statement should come before the LAG and DIF functions. For example, the following statements subset the FOREOUT data set shown in a previous example to select only _TYPE_=RESIDUAL observations and also to compute the variable LAGRESID.

```
data residual;
   set foreout;
   if _type_ = "RESIDUAL";
   lagresid = lag( cpi );
run;
```

Another pitfall of LAG and DIF functions arises when they are used to process time series cross-sectional data sets. For example, suppose you want to add the variable CPILAG to the CPICITY data set shown in a previous example. You might use the following statements:

```
data cpicity;
   set cpicity;
   cpilag = lag( cpi );
run;
```

However, these statements do not yield the desired result. In the data set produced by these statements, the value of CPILAG for the first observation for the first city is missing (as it should be), but in the first observation for all later cities, CPILAG contains the last value for the previous city. To correct this, set the lagged variable to missing at the start of each cross section, as follows.

```
data cpicity;
   set cpicity;
   by city date;
   cpilag = lag( cpi );
   if first.city then cpilag = .;
run;
```

Alternatives to LAG and DIF Functions

You can also calculate lags and differences in the DATA step without using LAG and DIF functions. For example, the following statements add the variables CPILAG and CPIDIF to the USCPI data set:

```
data uscpi;
   set uscpi;
   retain cpilag;
   cpidif = cpi - cpilag;
   output;
   cpilag = cpi;
run;
```

The RETAIN statement prevents the DATA step from reinitializing CPILAG to a missing value at the start of each iteration and thus allows CPILAG to retain the value of CPI assigned to it in the last statement. The OUTPUT statement causes the output observation to contain values of the variables before CPILAG is reassigned the current value of CPI in the last statement. This is the approach that must be used if you want to build a variable that is a function of its previous lags.

You can also use the EXPAND procedure to compute lags and differences. For example, the following statements compute lag and difference variables for CPI:

```
proc expand data=uscpi out=uscpi method=none;
   id date;
   convert cpi=cpilag / transform=( lag 1 );
   convert cpi=cpidif / transform=( dif 1 );
run;
```

LAG and DIF Functions in PROC MODEL

The preceding discussion of LAG and DIF functions applies to LAG and DIF functions available in the DATA step. However, LAG and DIF functions are also used in the MODEL procedure.

The MODEL procedure LAG and DIF functions do not work like the DATA step LAG and DIF functions. The LAG and DIF functions supported by PROC MODEL are true lag and difference functions, not queuing functions.

Unlike the DATA step, the MODEL procedure processes observations from a single input data set, so the "previous observation" is always clearly defined in a PROC MODEL program. Therefore, PROC MODEL is able to define LAG and DIF as true lagging functions that operate on values from the previous observation. See Chapter 20, “The MODEL Procedure,” for more information on LAG and DIF functions in the MODEL procedure.

Multiperiod Lags and Higher-Order Differencing

To compute lags at a lagging period greater than 1, add the lag length to the end of the LAG keyword to specify the lagging function needed. For example, the LAG2 function returns the value of its argument two calls ago, the LAG3 function returns the value of its argument three calls ago, and so forth.

To compute differences at a lagging period greater than 1, add the lag length to the end of the DIF keyword. For example, the DIF2 function computes the differences between the value of its argument and the value of its argument two calls ago. (The maximum lagging period is 100.)

The following statements add the variables CPILAG12 and CPIDIF12 to the USCPI data set. CPILAG12 contains the value of CPI from the same month one year ago. CPIDIF12 contains the change in CPI from the same month one year ago. (In this case, the first 12 values of CPILAG12 and CPIDIF12 will be missing.)

```
data uscpi;
   set uscpi;
   cpilag12 = lag12( cpi );
   cpidif12 = dif12( cpi );
run;
```

To compute second differences, take the difference of the difference. To compute higher-order differences, nest DIF functions to the order needed. For example, the following statements compute the second difference of CPI:

```
data uscpi;
   set uscpi;
   cpi2dif = dif( dif( cpi ) );
run;
```

Multiperiod lags and higher-order differencing can be combined. For example, the following statements compute monthly changes in the inflation rate, with inflation rate computed as percent change in CPI from the same month one year ago:

```
data uscpi;
   set uscpi;
   infchng = dif( 100 * dif12( cpi ) / lag12( cpi ) );
run;
```

Percent Change Calculations

There are several common ways to compute the percent change in a time series. This section illustrates the use of LAG and DIF functions by showing SAS statements for various kinds of percent change calculations.

Computing Period-to-Period Change

To compute percent change from the previous period, divide the difference of the series by the lagged value of the series and multiply by 100.

```
data uscpi;
   set uscpi;
   pctchng = dif( cpi ) / lag( cpi ) * 100;
   label pctchng = "Monthly Percent Change, At Monthly Rates";
run;
```

Often, changes from the previous period are expressed at annual rates. This is done by exponentiation of the current-to-previous period ratio to the number of periods in a year and expressing the result as a percent change. For example, the following statements compute the month-over-month change in CPI as a percent change at annual rates:

```
data uscpi;
   set uscpi;
   pctchng = ( ( cpi / lag( cpi ) ) ** 12 - 1 ) * 100;
   label pctchng = "Monthly Percent Change, At Annual Rates";
run;
```

Computing Year-over-Year Change

To compute percent change from the same period in the previous year, use LAG and DIF functions with a lagging period equal to the number of periods in a year. (For quarterly data, use LAG4 and DIF4. For monthly data, use LAG12 and DIF12.)

For example, the following statements compute monthly percent change in CPI from the same month one year ago:

```
data uscpi;
   set uscpi;
   pctchng = dif12( cpi ) / lag12( cpi ) * 100;
   label pctchng = "Percent Change from One Year Ago";
run;
```

To compute year-over-year percent change measured at a given period within the year, subset the series of percent changes from the same period in the previous year to form a yearly data set. Use an IF or WHERE statement to select observations for the period within each year on which the year-over-year changes are based.

For example, the following statements compute year-over-year percent change in CPI from December of the previous year to December of the current year:

```
data annual;
   set uscpi;
   pctchng = dif12( cpi ) / lag12( cpi ) * 100;
   label pctchng = "Percent Change: December to December";
   if month( date ) = 12;
   format date year4.;
run;
```

Computing Percent Change in Yearly Averages

To compute changes in yearly averages, first aggregate the series to an annual series using the EXPAND procedure, and then compute the percent change of the annual series. (See Chapter 16, “The EXPAND Procedure,” for more information on PROC EXPAND.)

For example, the following statements compute percent changes in the annual averages of CPI:

```
proc expand data=uscpi out=annual from=month to=year;
   convert cpi / observed=average method=aggregate;
run;

data annual;
   set annual;
   pctchng = dif( cpi ) / lag( cpi ) * 100;
   label pctchng = "Percent Change in Yearly Averages";
run;
```

It is also possible to compute percent change in the average over the most recent yearly span. For example, the following statements compute monthly percent change in the average of CPI over the most recent 12 months from the average over the previous 12 months:

```
data uscpi;
   retain sum12 0;
   drop sum12 ave12 cpilag12;
```

```
      set uscpi;
      sum12 = sum12 + cpi;
      cpilag12 = lag12( cpi );
      if cpilag12 ^= . then sum12 = sum12 - cpilag12;
      if lag11( cpi ) ^= . then ave12 = sum12 / 12;
      pctchng = dif12( ave12 ) / lag12( ave12 ) * 100;
      label pctchng = "Percent Change in 12 Month Moving Ave.";
   run;
```

This example is a complex use of LAG and DIF functions that requires care in handling the initialization of the moving-window averaging process. The LAG12 of CPI is checked for missing values to determine when more than 12 values have been accumulated, and older values must be removed from the moving sum. The LAG11 of CPI is checked for missing values to determine when at least 12 values have been accumulated; AVE12 will be missing when LAG11 of CPI is missing. The DROP statement prevents temporary variables from being added to the data set.

Note that the DIF and LAG functions must execute for every observation or the queues of remembered values will not operate correctly. The CPILAG12 calculation must be separate from the IF statement. The PCTCHNG calculation must not be conditional on the IF statement.

The EXPAND procedure provides an alternative way to compute moving averages.

Leading Series

Although the SAS System does not provide a function to look ahead at the "next" value of a series, there are a couple of ways to perform this task.

The most direct way to compute leads is to use the EXPAND procedure. For example

```
   proc expand data=uscpi out=uscpi method=none;
      id date;
      convert cpi=cpilead1 / transform=( lead 1 );
      convert cpi=cpilead2 / transform=( lead 2 );
   run;
```

Another way to compute lead series in SAS software is by lagging the time ID variable, renaming the series, and merging the result data set back with the original data set.

For example, the following statements add the variable CPILEAD to the USCPI data set. The variable CPILEAD contains the value of CPI in the following month. (The value of CPILEAD will be missing for the last observation, of course.)

```
   data temp;
      set uscpi;
      keep date cpi;
      rename cpi = cpilead;
      date = lag( date );
      if date ^= .;
```

```
run;

data uscpi;
   merge uscpi temp;
   by date;
run;
```

To compute leads at different lead lengths, you must create one temporary data set for each lead length. For example, the following statements compute CPILEAD1 and CPILEAD2, which contain leads of CPI for 1 and 2 periods, respectively:

```
data temp1(rename=(cpi=cpilead1)) temp2(rename=(cpi=cpilead2));
   set uscpi;
   keep date cpi;
   date = lag( date );
   if date ^= . then output temp1;
   date = lag( date );
   if date ^= . then output temp2;
run;

data uscpi;
   merge uscpi temp1 temp2;
   by date;
run;
```

Summing Series

Simple cumulative sums are easy to compute using SAS sum statements. The following statements show how to compute the running sum of variable X in data set A, adding XSUM to the data set.

```
data a;
   set a;
   xsum + x;
run;
```

The SAS sum statement automatically retains the variable XSUM and initializes it to 0, and the sum statement treats missing values as 0. The sum statement is equivalent to using a RETAIN statement and the SUM function. The previous example could also be written as follows:

```
data a;
   set a;
   retain xsum;
   xsum = sum( xsum, x );
run;
```

You can also use the EXPAND procedure to compute summations. For example

```
proc expand data=a out=a method=none;
   convert x=xsum / transform=( sum );
run;
```

Like differencing, summation can be done at different lags and can be repeated to produce higher-order sums. To compute sums over observations separated by lags greater than 1, use the LAG and SUM functions together, and use a RETAIN statement that initializes the summation variable to zero.

For example, the following statements add the variable XSUM2 to data set A. XSUM2 contains the sum of every other observation, with even-numbered observations containing a cumulative sum of values of X from even observations, and odd-numbered observations containing a cumulative sum of values of X from odd observations.

```
data a;
   set a;
   retain xsum2 0;
   xsum2 = sum( lag( xsum2 ), x );
run;
```

Assuming that A is a quarterly data set, the following statements compute running sums of X for each quarter. XSUM4 contains the cumulative sum of X for all observations for the same quarter as the current quarter. Thus, for a first-quarter observation, XSUM4 contains a cumulative sum of current and past first-quarter values.

```
data a;
   set a;
   retain xsum4 0;
   xsum4 = sum( lag3( xsum4 ), x );
run;
```

To compute higher-order sums, repeat the preceding process and sum the summation variable. For example, the following statements compute the first and second summations of X:

```
data a;
   set a;
   xsum + x;
   x2sum + xsum;
run;
```

The following statements compute the second order four-period sum of X:

```
data a;
   set a;
   retain xsum4 x2sum4 0;
   xsum4 = sum( lag3( xsum4 ), x );
   x2sum4 = sum( lag3( x2sum4 ), xsum4 );
run;
```

You can also use PROC EXPAND to compute cumulative statistics and moving window statistics. See Chapter 16, “The EXPAND Procedure,” for details.

Transforming Time Series

It is often useful to transform time series for analysis or forecasting. Many time series analysis and forecasting methods are most appropriate for time series with an unrestricted range, linear trend, and constant variance. Series that do not conform to these assumptions can often be transformed to series for which the methods are appropriate.

Transformations can be useful for the following:

- range restrictions. Many time series cannot have negative values or may be limited by a maximum possible value. You can often create a transformed series with an unbounded range.
- nonlinear trends. Many economic time series grow exponentially. Exponential growth corresponds to linear growth in the logarithms of the series.
- series variability that changes over time. Various transformations can be used to stabilize the variance.
- non-stationarity. The %DFTEST macro can be used to test a series for non-stationarity which may then be removed by differencing.

Log Transformation

The logarithmic transformation is often useful for series that must be greater than zero and that grow exponentially. For example, Figure 2.21 shows a plot of an airline passenger miles series. Notice that the series has exponential growth and the variability of the series increases over time. Airline passenger miles must also be zero or greater.

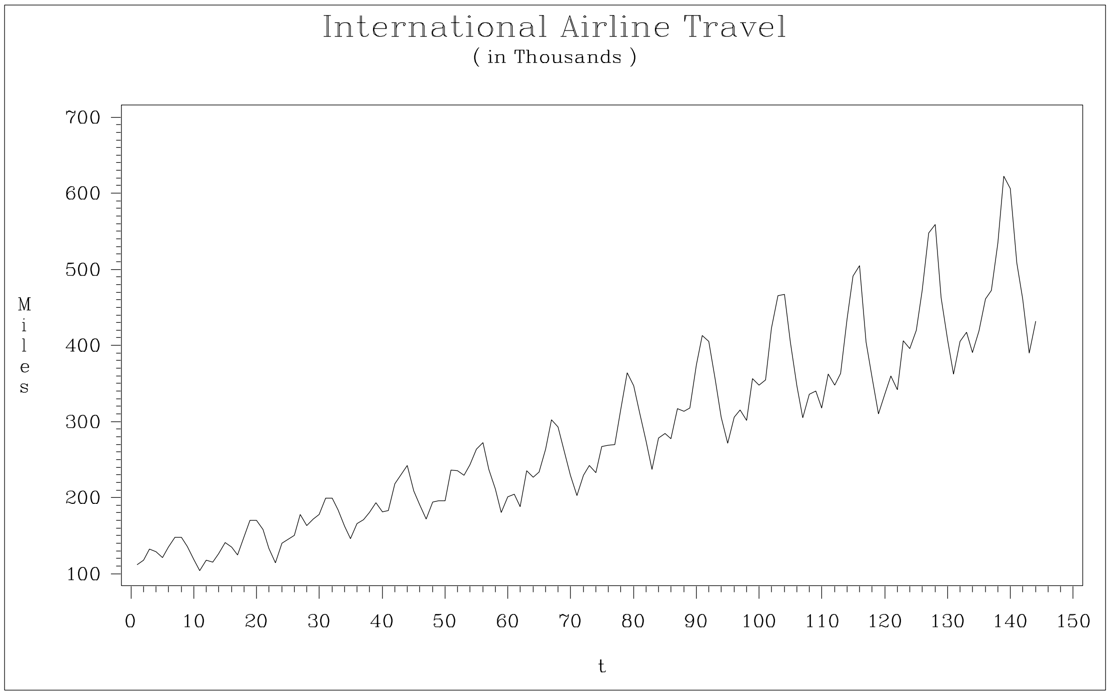


Figure 2.21. [Airline Series

The following statements compute the logarithms of the airline series:

```
data a;
   set a;
   logair = log( air );
run;
```

Figure 2.22 shows a plot of the log transformed airline series. Notice that the log series has a linear trend and constant variance.

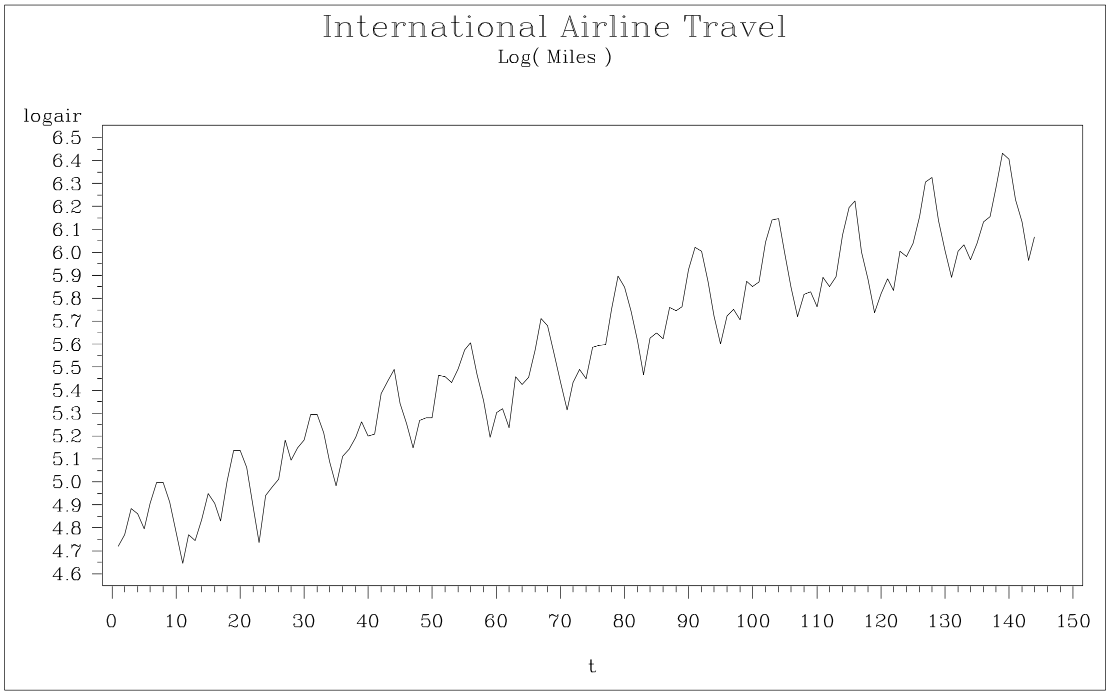


Figure 2.22. Log Airline Series

The %LOGTEST macro can help you decide if a log transformation is appropriate for a series. See Chapter 4, "SAS Macros and Functions," for more information on the %LOGTEST macro.

Other Transformations

The Box-Cox transformation is a general class of transformations that includes the logarithm as a special case. The %BOXCOXAR macro can be used to find an optimal Box-Cox transformation for a time series. See Chapter 4 for more information on the %BOXCOXAR macro.

The logistic transformation is useful for variables with both an upper and a lower bound, such as market shares. The logistic transformation is useful for proportions, percent values, relative frequencies, or probabilities. The logistic function transforms values between 0 and 1 to values that can range from $-\infty$ to $+\infty$.

For example, the following statements transform the variable SHARE from percent values to an unbounded range:

```
data a;
   set a;
   lshare = log( share / ( 100 - share ) );
run;
```

Many other data transformation can be used. You can create virtually any desired data transformation using DATA step statements.

The EXPAND Procedure and Data Transformations

The EXPAND procedure provides a convenient way to transform series. For example, the following statements add variables for the logarithm of AIR and the logistic of SHARE to data set A:

```
proc expand data=a out=a method=none;
   convert air=logair   / transform=( log );
   convert share=lshare / transform=( / 100 logit );
run;
```

See Table 16.1 in Chapter 16 for a complete list of transformations supported by PROC EXPAND.

Manipulating Time Series Data Sets

This section discusses merging, splitting, and transposing time series data sets and interpolating time series data to a higher or lower sampling frequency.

Splitting and Merging Data Sets

In some cases, you may want to separate several time series contained in one data set into different data sets. In other cases, you may want to combine time series from different data sets into one data set.

To split a time series data set into two or more data sets containing subsets of the series, use a DATA step to create the new data sets and use the KEEP= data set option to control which series are included in each new data set. The following statements split the USPRICE data set shown in a previous example into two data sets, USCPI and USPPI:

```
data uscpi(keep=date cpi)
     usppi(keep=date ppi);
   set usprice;
run;
```

If the series have different time ranges, you can subset the time ranges of the output data sets accordingly. For example, if you know that CPI in USPRICE has the range August 1990 through the end of the data set, while PPI has the range from the beginning of the data set through June 1991, you could write the previous example as follows:

```
data uscpi(keep=date cpi)
     usppi(keep=date ppi);
   set usprice;
   if date >= '1aug1990'd then output uscpi;
   if date <= '1jun1991'd then output usppi;
run;
```

To combine time series from different data sets into one data set, list the data sets to be combined in a MERGE statement and specify the dating variable in a BY statement. The following statements show how to combine the USCPI and USPPI data sets to produce the USPRICE data set. It is important to use the BY DATE; statement so observations are matched by time before merging.

```
data usprice;
   merge uscpi usppi;
   by date;
run;
```

Transposing Data Sets

The TRANSPOSE procedure is used to transpose data sets from one form to another. The TRANSPOSE procedure can transpose variables and observations, or transpose variables and observations within BY groups. This section discusses some applications of the TRANSPOSE procedure relevant to time series data sets. Refer to the *SAS Procedures Guide* for more information on PROC TRANSPOSE.

Transposing from Interleaved to Standard Time Series Form

The following statements transpose part of the interleaved form output data set FOREOUT, produced by PROC FORECAST in a previous example, to a standard form time series data set. To reduce the volume of output produced by the example, a WHERE statement is used to subset the input data set.

Observations with _TYPE_=ACTUAL are stored in the new variable ACTUAL; observations with _TYPE_=FORECAST are stored in the new variable FORECAST; and so forth. Note that the method used in this example only works for a single variable.

```
title "Original Data Set";
proc print data=foreout;
   where date > '1may1991'd & date < '1oct1991'd;
run;

proc transpose data=foreout out=trans(drop=_name_ _label_);
   var cpi;
   id _type_;
   by date;
   where date > '1may1991'd & date < '1oct1991'd;
run;

title "Transposed Data Set";
proc print data=trans;
run;
```

The TRANSPOSE procedure adds the variables _NAME_ and _LABEL_ to the output data set. These variables contain the names and labels of the variables that were transposed. In this example, there is only one transposed variable, so _NAME_ has the value CPI for all observations. Thus, _NAME_ and _LABEL_ are of no interest and are dropped from the output data set using the DROP= data set option. (If none of the variables transposed have a label, PROC TRANSPOSE does not output the _LABEL_ variable and the DROP=_LABEL_ option produces a warning message. You can ignore this message, or you can prevent the message by omitting _LABEL_ from the DROP= list.)

The original and transposed data sets are shown in Figure 2.23. (The observation numbers shown for the original data set reflect the operation of the WHERE statement.)

```
                          Original Data Set

             Obs       date     _TYPE_      _LEAD_         cpi

              37    JUN1991    ACTUAL         0       136.000
              38    JUN1991    FORECAST       0       136.146
              39    JUN1991    RESIDUAL       0        -0.146
              40    JUL1991    ACTUAL         0       136.200
              41    JUL1991    FORECAST       0       136.566
              42    JUL1991    RESIDUAL       0        -0.366
              43    AUG1991    FORECAST       1       136.856
              44    AUG1991    L95            1       135.723
              45    AUG1991    U95            1       137.990
              46    SEP1991    FORECAST       2       137.443
              47    SEP1991    L95            2       136.126
              48    SEP1991    U95            2       138.761
```

```
                         Transposed Data Set

   Obs       date     ACTUAL     FORECAST    RESIDUAL      L95        U95

    1     JUN1991     136.0      136.146    -0.14616       .          .
    2     JUL1991     136.2      136.566    -0.36635       .          .
    3     AUG1991        .       136.856      .         135.723    137.990
    4     SEP1991        .       137.443      .         136.126    138.761
```

Figure 2.23. Original and Transposed Data Sets

Transposing Cross-sectional Dimensions

The following statements transpose the variable CPI in the CPICITY data set shown in a previous example from time series cross-sectional form to a standard form time series data set. (Only a subset of the data shown in the previous example is used here.) Note that the method shown in this example only works for a single variable.

```
title "Original Data Set";
proc print data=cpicity;
run;

proc sort data=cpicity out=temp;
   by date city;
run;
```

```
proc transpose data=temp out=citycpi(drop=_name_ _label_);
   var cpi;
   id city;
   by date;
run;

title "Transposed Data Set";
proc print data=citycpi;
run;
```

The names of the variables in the transposed data sets are taken from the city names in the ID variable CITY. The original and the transposed data sets are shown in Figure 2.24.

```
                    Original Data Set

         Obs    city           date      cpi

           1    Chicago        JAN90    128.1
           2    Chicago        FEB90    129.2
           3    Chicago        MAR90    129.5
           4    Chicago        APR90    130.4
           5    Chicago        MAY90    130.4
           6    Chicago        JUN90    131.7
           7    Chicago        JUL90    132.0
           8    Los Angeles    JAN90    132.1
           9    Los Angeles    FEB90    133.6
          10    Los Angeles    MAR90    134.5
          11    Los Angeles    APR90    134.2
          12    Los Angeles    MAY90    134.6
          13    Los Angeles    JUN90    135.0
          14    Los Angeles    JUL90    135.6
          15    New York       JAN90    135.1
          16    New York       FEB90    135.3
          17    New York       MAR90    136.6
          18    New York       APR90    137.3
          19    New York       MAY90    137.2
          20    New York       JUN90    137.1
          21    New York       JUL90    138.4
```

```
                   Transposed Data Set

                                  Los_
    Obs     date     Chicago    Angeles    New_York

     1      JAN90     128.1      132.1       135.1
     2      FEB90     129.2      133.6       135.3
     3      MAR90     129.5      134.5       136.6
     4      APR90     130.4      134.2       137.3
     5      MAY90     130.4      134.6       137.2
     6      JUN90     131.7      135.0       137.1
     7      JUL90     132.0      135.6       138.4
```

Figure 2.24. Original and Transposed Data Sets

The following statements transpose the CITYCPI data set back to the original form of the CPICITY data set. The variable _NAME_ is added to the data set to tell PROC TRANSPOSE the name of the variable in which to store the observations in the transposed data set. (If the (DROP=_NAME_ _LABEL_) option were omitted from the first PROC TRANSPOSE step, this would not be necessary. PROC TRANSPOSE assumes ID _NAME_ by default.)

The NAME=CITY option in the PROC TRANSPOSE statement causes PROC TRANSPOSE to store the names of the transposed variables in the variable CITY. Because PROC TRANSPOSE recodes the values of the CITY variable to create valid SAS variable names in the transposed data set, the values of the variable CITY in the retransposed data set are not the same as the original. The retransposed data set is shown in Figure 2.25.

```
data temp;
   set citycpi;
   _name_ = 'CPI';
run;

proc transpose data=temp out=retrans name=city;
   by date;
run;

proc sort data=retrans;
   by city date;
run;

title "Retransposed Data Set";
proc print data=retrans;
run;
```

```
                    Retransposed Data Set

              Obs    date     city            CPI

                1    JAN90    Chicago        128.1
                2    FEB90    Chicago        129.2
                3    MAR90    Chicago        129.5
                4    APR90    Chicago        130.4
                5    MAY90    Chicago        130.4
                6    JUN90    Chicago        131.7
                7    JUL90    Chicago        132.0
                8    JAN90    Los_Angeles    132.1
                9    FEB90    Los_Angeles    133.6
               10    MAR90    Los_Angeles    134.5
               11    APR90    Los_Angeles    134.2
               12    MAY90    Los_Angeles    134.6
               13    JUN90    Los_Angeles    135.0
               14    JUL90    Los_Angeles    135.6
               15    JAN90    New_York       135.1
               16    FEB90    New_York       135.3
               17    MAR90    New_York       136.6
               18    APR90    New_York       137.3
               19    MAY90    New_York       137.2
               20    JUN90    New_York       137.1
               21    JUL90    New_York       138.4
```

Figure 2.25. Data Set Transposed Back to Original Form

Time Series Interpolation

The EXPAND procedure interpolates time series. This section provides a brief summary of the use of PROC EXPAND for different kinds of time series interpolation problems. Most of the issues discussed in this section are explained in greater detail in Chapter 16.

By default, the EXPAND procedure performs interpolation by first fitting cubic spline curves to the available data and then computing needed interpolating values from the fitted spline curves. Other interpolation methods can be requested.

Note that interpolating values of a time series does not add any real information to the data as the interpolation process is not the same process that generated the other (nonmissing) values in the series. While time series interpolation can sometimes be useful, great care is needed in analyzing time series containing interpolated values.

Interpolating Missing Values

To use the EXPAND procedure to interpolate missing values in a time series, specify the input and output data sets on the PROC EXPAND statement, and specify the time ID variable in an ID statement. For example, the following statements cause PROC EXPAND to interpolate values for missing values of all numeric variables in the data set USPRICE:

```
proc expand data=usprice out=interpl;
   id date;
run;
```

Interpolated values are computed only for embedded missing values in the input time series. Missing values before or after the range of a series are ignored by the EXPAND procedure.

In the preceding example, PROC EXPAND assumes that all series are measured at points in time given by the value of the ID variable. In fact, the series in the USPRICE data set are monthly averages. PROC EXPAND may produce a better interpolation if this is taken into account. The following example uses the FROM=MONTH option to tell PROC EXPAND that the series is monthly and uses the CONVERT statement with the OBSERVED=AVERAGE to specify that the series values are averages over each month:

```
proc expand data=usprice out=interpl from=month;
   id date;
   convert cpi ppi / observed=average;
run;
```

Interpolating to a Higher or Lower Frequency

You can use PROC EXPAND to interpolate values of time series at a higher or lower sampling frequency than the input time series. To change the periodicity of time series, specify the time interval of the input data set with the FROM= option, and specify the time interval for the desired output frequency with the TO= option. For example, the following statements compute interpolated weekly values of the monthly CPI and PPI series:

```
proc expand data=usprice out=interpl from=month to=week;
   id date;
   convert cpi ppi / observed=average;
run;
```

Interpolating between Stocks and Flows, Levels and Rates

A distinction is made between variables that are measured at points in time and variables that represent totals or averages over an interval. Point-in-time values are often called *stocks* or *levels*. Variables that represent totals or averages over an interval are often called *flows* or *rates*.

For example, the annual series Gross National Product represents the final goods production of over the year and also the yearly average rate of that production. However, the monthly variable Inventory represents the cost of a stock of goods at the end of the month.

The EXPAND procedure can convert between point-in-time values and period average or total values. To convert observation characteristics, specify the input and output characteristics with the OBSERVED= option in the CONVERT statement. For example, the following statements use the monthly average price index values in USPRICE to compute interpolated estimates of the price index levels at the midpoint of each month.

```
proc expand data=usprice out=midpoint from=month;
   id date;
   convert cpi ppi / observed=(average,middle);
run;
```

Reading Time Series Data

Time series data can be coded in many different ways. The SAS System can read time series data recorded in almost any form. Earlier sections of this chapter show how to read time series data coded in several commonly used ways. This section shows how to read time series data from data records coded in two other commonly used ways not previously introduced.

Several time series databases distributed by major data vendors can be read into SAS data sets by the DATASOURCE procedure. See Chapter 14, “The DATASOURCE Procedure,” for more information.

The SASECRSP, SASEFAME, and SASEHAVR interface engines enables SAS users to access and process time series data in CRSPAccess data files, FAME databases, and HAVER ANALYTICS Data Link Express (DLX) data bases, respectively. See

Chapter 5, “The SASECRSP Interface Engine,” Chapter 6, “The SASEFAME Interface Engine,” and Chapter 7, “The SASEHAVR Interface Engine,” for more details.

Reading a Simple List of Values

Time series data can be coded as a simple list of values without dating information and with an arbitrary number of observations on each data record. In this case, the INPUT statement must use the trailing "@@" option to retain the current data record after reading the values for each observation, and the time ID variable must be generated with programming statements.

For example, the following statements read the USPRICE data set from data records containing pairs of values for CPI and PPI. This example assumes you know that the first pair of values is for June 1990.

```
data usprice;
   input cpi ppi @@;
   date = intnx( ’month’, ’1jun1990’d, _n_-1 );
   format date monyy7.;
datalines;
129.9 114.3  130.4 114.5  131.6 116.5
132.7 118.4  133.5 120.8  133.8 120.1 133.8 118.7
134.6 119.0  134.8 117.2  135.0 116.2 135.2 116.0
135.6 116.5  136.0 116.3  136.2 116.0
;
```

Reading Fully Described Time Series in Transposed Form

Data for several time series can be coded with separate groups of records for each time series. Data files coded this way are transposed from the form required by SAS procedures. Time series data can also be coded with descriptive information about the series included with the data records.

The following example reads time series data for the USPRICE data set coded with separate groups of records for each series. The data records for each series consist of a series description record and one or more value records. The series description record gives the series name, starting month and year of the series, number of values in the series, and a series label. The value records contain the observations of the time series.

The data are first read into a temporary data set that contains one observation for each value of each series. This data set is sorted by date and series name, and the TRANSPOSE procedure is used to transpose the data into a standard form time series data set.

```
data temp;
   length _name_ $8 _label_ $40;
   keep _name_ _label_ date value;
   format date monyy.;
```

```
      input _name_ month year nval _label_ &;
      date = mdy( month, 1, year );
      do i = 1 to nval;
         input value @;
         output;
         date = intnx( 'month', date, 1 );
      end;
   datalines;
   cpi      8 90  12  Consumer Price Index
   131.6 132.7 133.5 133.8 133.8 134.6 134.8 135.0
   135.2 135.6 136.0 136.2
   ppi      6 90  13  Producer Price Index
   114.3 114.5 116.5 118.4 120.8 120.1 118.7 119.0
   117.2 116.2 116.0 116.5 116.3
   ;

   proc sort data=temp;
      by date _name_;
   run;

   proc transpose data=temp out=usprice(drop=_name_ _label_);
      by date;
      var value;
   run;

   proc contents data=usprice;
   run;

   proc print data=usprice;
   run;
```

The final data set is shown in Figure 2.26.

```
                            The CONTENTS Procedure

   Data Set Name: WORK.USPRICE                        Observations:         14
   Member Type:   DATA                                Variables:            3
   Engine:        V8                                  Indexes:              0
   Created:       17:38 Monday, May 3, 1999           Observation Length:   24
   Last Modified: 17:38 Monday, May 3, 1999           Deleted Observations: 0
   Protection:                                        Compressed:           NO
   Data Set Type:                                     Sorted:               NO
   Label:


             -----Alphabetic List of Variables and Attributes-----

     #    Variable    Type    Len    Pos    Format    Label
     ---------------------------------------------------------------------
     3    cpi         Num       8     16              Consumer Price Index
     1    date        Num       8      0    MONYY.
     2    ppi         Num       8      8              Producer Price Index
```

```
                          Obs    date      ppi      cpi

                            1    JUN90    114.3      .
                            2    JUL90    114.5      .
                            3    AUG90    116.5    131.6
                            4    SEP90    118.4    132.7
                            5    OCT90    120.8    133.5
                            6    NOV90    120.1    133.8
                            7    DEC90    118.7    133.8
                            8    JAN91    119.0    134.6
                            9    FEB91    117.2    134.8
                           10    MAR91    116.2    135.0
                           11    APR91    116.0    135.2
                           12    MAY91    116.5    135.6
                           13    JUN91    116.3    136.0
                           14    JUL91      .      136.2
```

Figure 2.26. USPRICE Data Set

Chapter 3
Date Intervals, Formats, and Functions

Chapter Contents

Chapter 3
Date Intervals, Formats, and Functions

Overview

This chapter summarizes the time intervals, date and datetime informats, date and datetime formats, and date, time and datetime functions available in the SAS System. The use of these features is explained in Chapter 2, "Working with Time Series Data." The material in this chapter is also contained in the *SAS Language: Reference*. Because these features are useful for work with time series data, documentation of these features is consolidated and repeated here for easy reference.

Time Intervals

This section provides a reference for the different kinds of time intervals supported by the SAS System. How intervals are used is not discussed here; see Chapter 2, "Working with Time Series Data," for an introduction to the use of time intervals.

Some interval names are for use with SAS date values, while other interval names are for use with SAS datetime values. The interval names used with SAS date values are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, and DAY. The interval names used with SAS datetime or time values are HOUR, MINUTE, and SECOND. Various abbreviations of these names are also allowed, as described in the section "Summary of Interval Types."

Interval names for use with SAS date values can be prefixed with 'DT' to construct interval names for use with SAS datetime values. The interval names DTYEAR, DTSEMIYEAR, DTQTR, DTMONTH, DTSEMIMONTH, DTTENDAY, DTWEEK, DTWEEKDAY, and DTDAY are used with SAS datetime or time values.

Constructing Interval Names

Multipliers and shift indexes can be used with the basic interval names to construct more complex interval specifications. The general form of an interval name is as follows:

NAMEn.s

The three parts of the interval name are:

NAME	the name of the basic interval type. For example, YEAR specifies yearly intervals.
n	an optional multiplier that specifies that the interval is a multiple of the period of the basic interval type. For example, the interval YEAR2 consists of two-year, or biennial, periods.
s	an optional starting subperiod index that specifies that the intervals are shifted to later starting points. For example, YEAR.3 specifies yearly periods shifted to start on the first of March of each calendar year and to end in February of the following year.

Both the multiplier *n* and the shift index *s* are optional and default to 1. For example, YEAR, YEAR1, YEAR.1, and YEAR1.1 are all equivalent ways of specifying ordinary calendar years.

Both the multiplier *n* and the shift index *s* are optional and default to 1. For example, YEAR, YEAR1, YEAR.1, and YEAR1.1 are all equivalent ways of specifying ordinary calendar years.

Shifted Intervals

Different kinds of intervals are shifted by different subperiods.

- YEAR, SEMIYEAR, QTR, and MONTH intervals are shifted by calendar months.
- WEEK, WEEKDAY, and DAY intervals are shifted by days.
- SEMIMONTH intervals are shifted by semi-monthly periods.
- TENDAY intervals are shifted by ten-day periods.
- HOUR intervals are shifted by hours.
- MINUTE intervals are shifted by minutes.
- SECOND intervals are shifted by seconds.

If a subperiod is specified, the shift index cannot be greater than the number of subperiods in the whole interval. For example, you could use YEAR2.24, but YEAR2.25 would be an error because there is no twenty-fifth month in a two-year interval. For interval types that shift by subperiods that are the same as the basic interval type, only multiperiod intervals can be shifted.

For example, MONTH type intervals shift by MONTH subintervals; thus, monthly intervals cannot be shifted since there is only one month in MONTH. However, bimonthly intervals can be shifted, since there are two MONTH intervals in each MONTH2 interval. The interval name MONTH2.2 specifies bimonthly periods starting on the first day of even-numbered months.

Alignment of Intervals

Intervals that represent divisions of a year are aligned with the start of the year (January). MONTH2 periods begin with odd-numbered months (January, March, May, and so on). Likewise, intervals that represent divisions of a day are aligned with the start of the day (midnight). Thus, HOUR8.7 intervals divide the day into the periods 06:00 to 14:00, 14:00 to 22:00, and 22:00 to 06:00.

Intervals that do not nest within years or days are aligned relative to the SAS date or datetime value 0. The arbitrary reference time of midnight on January 1, 1960, is used as the origin for nonshifted intervals, and shifted intervals are defined relative to that reference point. For example, MONTH13 defines the intervals January 1, 1960, February 1, 1961, March 1, 1962, and so forth, and the intervals December 1, 1959, November 1, 1958, and so on before the base date January 1, 1960.

Similarly, WEEK2 interval beginning days are aligned relative to the Sunday of the week of January 1, 1960. The interval specification WEEK6.13 defines six-week periods starting on second Fridays, and the convention of alignment relative to the period containing January 1, 1960, tells where to start counting to find out what dates correspond to the second Fridays of six-week intervals.

See the section “Alignment of SAS Dates” later in this chapter.

Summary of Interval Types

The interval types are summarized as follows.

YEAR
specifies yearly intervals. Abbreviations are YEAR, YEARS, YEARLY, YR, ANNUAL, ANNUALLY, ANNUALS. The starting subperiod *s* is in months.

SEMIYEAR
specifies semiannual intervals (every six months). Abbreviations are SEMIYEAR, SEMIYEARS, SEMIYEARLY, SEMIYR, SEMIANNUAL, SEMIANN.

The starting subperiod *s* is in months. For example, SEMIYEAR.3 intervals are March–August and September–February.

QTR
specifies quarterly intervals (every three months). Abbreviations are QTR, QUARTER, QUARTERS, QUARTERLY, QTRLY, QTRS. The starting subperiod *s* is in months.

MONTH
specifies monthly intervals. Abbreviations are MONTH, MONTHS, MONTHLY, MON.

The starting subperiod *s* is in months. For example, MONTH2.2 intervals are February–March, April–May, June–July, August–September, October–November, and December–January of the following year.

SEMIMONTH
specifies semimonthly intervals. SEMIMONTH breaks each month into two pe-

riods, starting on the first and sixteenth day. Abbreviations are SEMIMONTH, SEMIMONTHS, SEMIMONTHLY, SEMIMON.

The starting subperiod *s* is in SEMIMONTH periods. For example, SEMIMONTH2.2 specifies intervals from the sixteenth of one month through the fifteenth of the next month.

TENDAY

specifies 10-day intervals. TENDAY breaks the month into three periods, the first through the tenth day of the month, the eleventh through the twentieth day of the month, and the remainder of the month. (TENDAY is a special interval typically used for reporting automobile sales data.)

The starting subperiod *s* is in TENDAY periods. For example, TENDAY4.2 defines 40-day periods starting at the second TENDAY period.

WEEK

specifies weekly intervals of seven days. Abbreviations are WEEK, WEEKS, WEEKLY.

The starting subperiod *s* is in days, with the days of the week numbered as 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, and 7=Saturday. For example, WEEK.7 means weekly with Saturday as the first day of the week.

WEEKDAY
WEEKDAY17W

specifies daily intervals with weekend days included in the preceding week day. Abbreviations are WEEKDAY, WEEKDAYS.

The WEEKDAY interval is the same as DAY except that weekend days are absorbed into the preceding weekday. Thus there are five WEEKDAY intervals in a calendar week: Monday, Tuesday, Wednesday, Thursday, and the three-day period Friday-Saturday-Sunday.

The default weekend days are Saturday and Sunday, but any one to six weekend days can be listed after the WEEKDAY string and followed by a W. Weekend days are specified as ’1’ for Sunday, ’2’ for Monday, and so forth. For example, WEEKDAY67W specifies a Friday-Saturday weekend. WEEKDAY1W specifies a six-day work week with a Sunday weekend. WEEKDAY17W is the same as WEEKDAY.

The starting subperiod *s* is in days.

DAY

specifies daily intervals. Abbreviations are DAY, DAYS, DAILY. The starting subperiod *s* is in days.

HOUR

specifies hourly intervals. Abbreviations are HOUR, HOURS, HOURLY, HR. The starting subperiod *s* is in hours.

MINUTE

specifies minute intervals. Abbreviations are MINUTE, MINUTES, MIN. The starting subperiod *s* is in minutes.

SECOND
specifies second intervals. Abbreviations are SECOND, SECONDS, SEC. The starting subperiod *s* is in seconds.

Examples of Interval Specifications

Table 3.1 shows examples of different kinds of interval specifications.

Table 3.1. Examples of Intervals

Name	Kind of Interval
YEAR	years starting in January
YEAR.10	fiscal years starting in October
YEAR2.7	biennial intervals starting in July of even years
YEAR2.19	biennial intervals starting in July of odd years
YEAR4.11	four-year intervals starting in November of leap years (frequency of U.S. presidential elections)
YEAR4.35	four-year intervals starting in November of even years between leap years (frequency of U.S. midterm elections)
WEEK	weekly intervals starting on Sundays
WEEK2	biweekly intervals starting on first Sundays
WEEK1.1	same as WEEK
WEEK.2	weekly intervals starting on Mondays
WEEK6.3	six-week intervals starting on first Tuesdays
WEEK6.11	six-week intervals starting on second Wednesdays
WEEKDAY	daily with Friday-Saturday-Sunday counted as the same day (five-day work week with a Saturday-Sunday weekend)
WEEKDAY17W	same as WEEKDAY
WEEKDAY67W	daily with Thursday-Friday-Saturday counted as the same day (five-day work week with a Friday-Saturday weekend)
WEEKDAY1W	daily with Saturday-Sunday counted as the same day (six-day work week with a Sunday weekend)
WEEKDAY3.2	three-weekday intervals (with Friday-Saturday-Sunday counted as one weekday) with the cycle three-weekday periods aligned to Monday 4 Jan 1960
HOUR8.7	eight-hour intervals starting at 6 a.m., 2 p.m., and 10 p.m. (might be used for work shifts)

Date and Datetime Informats

Table 3.2 summarizes the SAS date and datetime informats available in the SAS System. See Chapter 2, "Working with Time Series Data," for a discussion of the use of date and datetime informats. Refer to *SAS Language: Reference* for a complete description of these informats.

For each informat, Table 3.2 shows an example of a date or datetime value written in the style that the informat is designed to read. The date 17 October 1991 and the time

2:25:32 p.m. are used for the example in all cases. Table 3.2 shows the width range allowed by the informat and the default width.

Table 3.2. SAS Date and Datetime Informats

Informat **Example**	**Description**	**Width Range**	**Default Width**
DATE*w.* 17oct91	day, month abbreviation, and year: *ddMONyy*	7-32	7
DATETIME*w.d* 17oct91:14:45:32	date and time: *ddMONyy:hh:mm:ss*	13-40	18
DDMMYY*w.* 17/10/91	day, month, year: *ddmmyy*, *dd/mm/yy*, *dd-mm-yy*, or *dd mm yy*	6-32	6
JULIAN*w.* 91290	year and day of year (Julian dates): *yyddd*	5-32	5
MMDDYY*w.* 10/17/91	month, day, year: *mmddyy*, *mm/dd/yy*, *mm-dd-yy*, or *mm dd yy*	6-32	6
MONYY*w.* Oct91	month abbreviation and year	5-32	5
NENGO*w.* H.03/10/17	Japanese Nengo notation	7-32	10
TIME*w.d* 14:45:32	hours, minutes, seconds: *hh:mm:ss* or hours, minutes: *hh:mm*.	5-32	8
YYMMDD*w.* 91/10/17	year, month, day: *yymmdd*, *yy/mm/dd*, *yy-mm-dd*, or *yy mm dd*	6-32	6
YYQ*w.* 91Q4	year and quarter of year: *yyQq*	4-32	4

Date, Time, and Datetime Formats

The SAS date and datetime formats are summarized in Table 3.3 and Table 3.4. A width value can be specified with each format. The tables list the range of width values allowed and the default width value for each format.

The notation used by a format is abbreviated in different ways depending on the width option used. For example, the format MMDDYY8. writes the date 17 October 1991 as 10/17/91, while the format MMDDYY6. writes this date as 101791. In particular, formats that display the year show two- or four-digit year values depending on the width option. The examples shown in the tables are for the default width.

Refer to *SAS Language: Reference* for a complete description of these formats, including the variations of the formats produced by different width options. See Chapter 2, "Working with Time Series Data," for a discussion of the use of date and datetime formats.

Date Formats

Table 3.3 lists the date formats available in the SAS System. For each format, an example is shown of a date value in the notation produced by the format. The date '17OCT91'D is used as the example.

Table 3.3. SAS Date Formats

Format **Example**	**Description**	**Width Range**	**Default Width**
DATE*w.* 17oct91	day, month abbreviation, year: *ddMONyy*	5-9	7
DAY*w.* 17	day of month	2-32	2
DDMMYY*w.* 17/10/91	day, month, year: *dd/mm/yy*	2-8	8
DOWNAME*w.* Thursday	name of day of the week	1-32	9
JULDAY*w.* 290	day of year	3-32	3
JULIAN*w.* 91290	year and day of year: *yyddd*	5-7	5
MMDDYY*w.* 10/17/91	month, day, year: *mm/dd/yy*	2-8	8
MMYY*w.* 10M1991	month and year: *mmMyy*	5-32	7
MMYYC*w.* 10:1991	month and year: *mm:yy*	5-32	7
MMYYD*w.* 10-1991	month and year: *mm-yy*	5-32	7
MMYYP*w.*	month and year: *mm.yy*	5-32	7

Table 3.3. (continued)

Format Example	**Description**	**Width Range**	**Default Width**
10.1991			
MMYYS*w.* 10/1991	month and year: *mm/yy*	5-32	7
MMYYN*w.* 101991	month and year: *mmyy*	5-32	6
MONNAME*w.* October	name of month	1-32	9
MONTH*w.* 10	month of year	1-32	2
MONYY*w.* OCT91	month abbreviation and year: *MONyy*	5-7	5
QTR*w.* 4	quarter of year	1-32	1
QTRR*w.* IV	quarter in Roman numerals	3-32	3
NENGO*w.* H.03/10/17	Japanese Nengo notation	2-10	10
WEEKDATE*w.* Thursday, October 17, 1991	*day-of-week, month-name dd, yy*	3-37	29
WEEKDATX*w.* Thursday, 17 October 1991	*day-of-week, dd month-name yy*	3-37	29
WEEKDAY*w.* 5	day of week	1-32	1
WORDDATE*w.* October 17, 1991	*month-name dd, yy*	3-32	18
WORDDATX*w.* 17 October 1991	*dd month-name yy*	3-32	18
YEAR*w.* 1991	year	2-32	4

Table 3.3. (continued)

Format **Example**	**Description**	**Width Range**	**Default Width**
YYMM*w*. 1991M10	year and month: *yyMmm*	5-32	7
YYMMC*w*. 1991:10	year and month: *yy:mm*	5-32	7
YYMMD*w*. 1991-10	year and month: *yy-mm*	5-32	7
YYMMP*w*. 1991.10	year and month: *yy.mm*	5-32	7
YYMMS*w*. 1991/10	year and month: *yy/mm*	5-32	7
YYMMN*w*. 199110	year and month: *yymm*	5-32	7
YYMON*w*. 1991OCT	year and month abbreviation: *yyMON*	5-32	7
YYMMDD*w*. 91/10/17	year, month, day: *yy/mm/dd*	2-8	8
YYQ*w*. 91Q4	year and quarter: *yyQq*	4-6	4
YYQC*w*. 1991:4	year and quarter: *yy:q*	4-32	6
YYQD*w*. 1991-4	year and quarter: *yy-q*	4-32	6
YYQP*w*. 1991.4	year and quarter: *yy.q*	4-32	6
YYQS*w*. 1991/4	year and quarter: *yy/q*	4-32	6
YYQN*w*. 19914	year and quarter: *yyq*	3-32	5
YYQR*w*. 1991QIV	year and quarter in Roman numerals: *yyQrr*	6-32	8

Table 3.3. (continued)

Format **Example**	**Description**	**Width Range**	**Default Width**
YYQRC*w.* 1991:IV	year and quarter in Roman numerals: *yy:rr*	6-32	8
YYQRD*w.* 1991-IV	year and quarter in Roman numerals: *yy-rr*	6-32	8
YYQRP*w.* 1991.IV	year and quarter in Roman numerals: *yy.rr*	6-32	8
YYQRS*w.* 1991/IV	year and quarter in Roman numerals: *yy/rr*	6-32	8
YYQRN*w.* 1991IV	year and quarter in Roman numerals: *yyrr*	6-32	8

Datetime and Time Formats

Table 3.4 lists the datetime and time formats available. For each format, an example is shown of a datetime value in the notation produced by the format. The datetime value ’17OCT91:14:25:32’DT is used as the example.

Table 3.4. SAS Datetime and Time Formats

Format **Example**	**Description**	**Width Range**	**Default Width**
DATETIME*w.d* 17OCT91:14:25:32	*ddMONyy:hh:mm:ss*	7-40	16
HHMM*w.d* 14:25	hour and minute: *hh:mm*	2-20	5
HOUR*w.d* 14	hour	2-20	2
MMSS*w.d* 25:32	minutes and seconds: *mm:ss*	2-20	5
TIME*w.d* 14:25:32	time of day: *hh:mm:ss*	2-20	8
TOD*w.* 14:25:32	time of day: *hh:mm:ss*	2-20	8

Alignment of SAS Dates

SAS date values used to identify time series observations produced by SAS/ETS procedures are normally aligned with the beginning of the time intervals corresponding to the observations. For example, for monthly data for 1994, the date values identifying the observations are 1Jan94, 1Feb94, 1Mar94, . . . , 1Dec94.

However, for some applications it may be preferable to use end of period dates, such as 31Jan94, 28Feb94, 31Mar94, . . . , 31Dec94. For other applications, such as plotting time series, it may be more convenient to use interval midpoint dates to identify the observations.

SAS/ETS procedures provide an ALIGN= option to control the alignment of dates for output time series observations. Procedures supporting the ALIGN= option are ARIMA, DATASOURCE, EXPAND, and FORECAST.

ALIGN=

The ALIGN= option allows the following values:

BEGINNING	Specifies that dates are aligned to the start of the interval. This is the default. BEGINNING can be abbreviated as BEGIN, BEG, or B.
MIDDLE	Specifies that dates are aligned to the interval midpoint. MIDDLE can be abbreviated as MID or M.
ENDING	Specifies that dates are aligned to the end of the interval. ENDING can be abbreviated as END or E.

The ALIGN= option can be specified on the PROC DATASOURCE statement, on the PROC EXPAND statement, on the PROC FORECAST statement, and on the FORECAST statement of the ARIMA procedure.

Date, Time, and Datetime Functions

The SAS System provides functions to perform calculations with SAS date, time, and datetime values. SAS date, time, and datetime functions are used to:

- compute date, time, and datetime values from calendar and time-of-day values.
- compute calendar and time-of-day values from date and datetime values.
- convert between date, time, and datetime values.
- perform calculations involving time intervals.

SAS date, time, and datetime functions are listed in alphabetical order in the following. Refer to *SAS Language: Reference* for a complete description of these functions.

SAS Date, Time, and Datetime Functions

DATE()
returns today's date as a SAS date value.

DATEJUL(*yyddd* **)**
returns the Julian date for a SAS date value.

DATEPART(*datetime* **)**
returns the date part of a SAS datetime value as a date value.

DATETIME()
returns the current date and time of day.

DAY(*date* **)**
returns the day of the month from a SAS date value.

DHMS(*date, hour, minute, second* **)**
returns a SAS datetime value for date, hour, minute, and second values.

HMS(*hour, minute, second* **)**
returns a SAS time value for hour, minute, and second values.

HOUR(*datetime* **)**
returns the hour from a SAS datetime or time value.

INTCINDEX(*'interval', value* **)**
returns the cycle index, given a date, time, or datetime interval and value.

INTCK(*interval, date1, date2* **)**
returns the number of boundaries of intervals of the given kind that lie between the two date or datetime values.

INTCYCLE(*'interval'* **)**
returns the date, time, or datetime interval at the next higher seasonal cycle, given a date, time, or datetime interval.

INTFMT(*'interval', 'size'* **)**
returns a recommended format, given a date, time, or datetime interval.

INTINDEX(*'interval', value* **)**
returns the seasonal index, given a date, time, or datetime interval and value.

INTNX(*interval, date, n* **<**, *'alignment'* **>)**
returns the date or datetime value of the beginning of the interval that is *n* intervals from the interval that contains the given date or datetime value. The optional alignment argument specifies that the returned date is aligned to either the beginning, middle, or end of the interval. Beginning is the default.

INTSEA(*'interval'* **)**
returns the length of the seasonal cycle, given a date, time, or datetime interval.

JULDATE(*date* **)**
returns the Julian date from a SAS date value.

MDY(*month, day, year* **)**
returns a SAS date value for month, day, and year values.

MINUTE(*datetime* **)**
returns the minute from a SAS time or datetime value.

MONTH(*date* **)**
returns the month of the year from a SAS date value.

QTR(*date* **)**
returns the quarter of the year from a SAS date value.

SECOND(*date* **)**
returns the second from a SAS time or datetime value.

TIME()
returns the current time of day.

TIMEPART(*datetime* **)**
returns the time part of a SAS datetime value.

TODAY()
returns the current date as a SAS date value. (TODAY is another name for the DATE function.)

WEEKDAY(*date* **)**
returns the day of the week from a SAS date value.

YEAR(*date* **)**
returns the year from a SAS date value.

YYQ(*year, quarter* **)**
returns a SAS date value for year and quarter values.

Chapter 4
SAS Macros and Functions

Chapter Contents

Chapter 4
SAS Macros and Functions

SAS Macros

This chapter describes several SAS macros and the SAS function PROBDF that are provided with SAS/ETS software. A SAS macro is a program that generates SAS statements. Macros make it easy to produce and execute complex SAS programs that would be time-consuming to write yourself.

SAS/ETS software includes the following macros:

%AR	generates statements to define autoregressive error models for the MODEL procedure.
%BOXCOXAR	investigates Box-Cox transformations useful for modeling and forecasting a time series.
%DFPVALUE	computes probabilities for Dickey-Fuller test statistics.
%DFTEST	performs Dickey-Fuller tests for unit roots in a time series process.
%LOGTEST	tests to see if a log transformation is appropriate for modeling and forecasting a time series.
%MA	generates statements to define moving average error models for the MODEL procedure.
%PDL	generates statements to define polynomial distributed lag models for the MODEL procedure.

These macros are part of the SAS AUTOCALL facility and are automatically available for use in your SAS program. Refer to *SAS Macro Language: Reference* for information about the SAS macro facility.

Since the %AR, %MA, and %PDL macros are used only with PROC MODEL, they are documented with the MODEL procedure. See the sections on the %AR, %MA, and %PDL macros in Chapter 20, "The MODEL Procedure," for more information about these macros. The %BOXCOXAR, %DFPVALUE, %DFTEST, and %LOGTEST macros are described in the following sections.

BOXCOXAR Macro

The %BOXCOXAR macro finds the optimal Box-Cox transformation for a time series.

Transformations of the dependent variable are a useful way of dealing with nonlinear relationships or heteroscedasticity. For example, the logarithmic transformation is often used for modeling and forecasting time series that show exponential growth or that show variability proportional to the level of the series.

The Box-Cox transformation is a general class of power transformations that include the log transformation and no-transformation as special cases. The Box-Cox transformation is

$$Y_t = \begin{cases} \frac{(X_t+c)^\lambda - 1}{\lambda} & \text{for } \lambda \neq 0 \\ \ln(X_t + c) & \text{for } \lambda = 0 \end{cases}$$

The parameter λ controls the shape of the transformation. For example, λ=0 produces a log transformation, while λ=.5 results in a square root transformation. When λ=1 the transformed series differs from the original series by $c - 1$.

The constant *c* is optional. It can be used when some X_t values are negative or 0. You choose *c* so that the series X_t is always greater than $-c$.

The %BOXCOXAR macro tries a range of λ values and reports which of the values tried produces the optimal Box-Cox transformation. To evaluate different λ values, the %BOXCOXAR macro transforms the series with each λ value and fits an autoregressive model to the transformed series. It is assumed that this autoregressive model is a reasonably good approximation to the true time series model appropriate for the transformed series. The likelihood of the data under each autoregressive model is computed, and the λ value producing the maximum likelihood over the values tried is reported as the optimal Box-Cox transformation for the series.

The %BOXCOXAR macro prints and optionally writes to a SAS data set all of the λ values tried and the corresponding log likelihood value and related statistics for the autoregressive model.

You can control the range and number of λ values tried. You can also control the order of the autoregressive models fit to the transformed series. You can difference the transformed series before the autoregressive model is fit.

Syntax

The form of the %BOXCOXAR macro is

%BOXCOXAR *(SAS-data-set, variable [, options])*

The first argument, *SAS-data-set*, specifies the name of the SAS data set containing the time series to be analyzed. The second argument, *variable*, specifies the time series variable name to be analyzed. The first two arguments are required.

The following options can be used with the %BOXCOXAR macro. Options must follow the required arguments and are separated by commas.

AR= *n*
specifies the order of the autoregressive model fit to the transformed series. The default is AR=5.

CONST= *value*
specifies a constant *c* to be added to the series before transformation. Use the CONST= option when some values of the series are 0 or negative. The default is CONST=0.

DIF= *(differencing-list)*
specifies the degrees of differencing to apply to the transformed series before the autoregressive model is fit. The *differencing-list* is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the transformed series be differenced once at lag 1 and once at lag 12. For more details, see "IDENTIFY Statement" in Chapter 11, "The ARIMA Procedure."

LAMBDAHI= *value*
specifies the maximum value of lambda for the grid search. The default is LAMBDAHI=1. A large (in magnitude) LAMBDAHI= value can result in problems with floating point arithmetic.

LAMBDALO= *value*
specifies the minimum value of lambda for the grid search. The default is LAMBDALO=0. A large (in magnitude) LAMBDALO= value can result in problems with floating point arithmetic.

NLAMBDA= *value*
specifies the number of lambda values considered, including the LAMBDALO= and LAMBDAHI= option values. The default is NLAMBDA=2.

OUT= *SAS-data-set*
writes the results to an output data set. The output data set includes the lambda values tried (LAMBDA), and for each lambda value the log likelihood (LOGLIK), residual mean square error (RMSE), Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC).

PRINT= YES | NO
specifies whether results are printed. The default is PRINT=YES. The printed output contains the lambda values, log likelihoods, residual mean square errors, Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC).

Results

The value of λ producing the maximum log likelihood is returned in the macro variable &BOXCOXAR. The value of the variable &BOXCOXAR is "ERROR" if the %BOXCOXAR macro is unable to compute the best transformation due to errors. This may be the result of large lambda values. The Box-Cox transformation parameter involves exponentiation of the data, so that large lambda values may cause floating-point overflow.

Results are printed unless the PRINT=NO option is specified. Results are also stored in SAS data sets when the OUT= option is specified.

Details

Assume that the transformed series Y_t is a stationary *p*th order autoregressive process generated by independent normally distributed innovations.

$$(1 - \Theta(B))(Y_t - \mu) = \epsilon_t$$

$$\epsilon_t \sim iid\mathrm{N}(0, \sigma^2)$$

Given these assumptions, the log likelihood function of the transformed data Y_t is

$$\begin{aligned} l_Y(\cdot) = & - \frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma|) - \frac{n}{2}\ln(\sigma^2) \\ & - \frac{1}{2\sigma^2}(\mathbf{Y} - \mathbf{1}\mu)'\Sigma^{-1}(\mathbf{Y} - \mathbf{1}\mu) \end{aligned}$$

In this equation, *n* is the number of observations, μ is the mean of Y_t, **1** is the *n*-dimensional column vector of 1s, σ^2 is the innovation variance, $\mathbf{Y} = (Y_1, \cdots, Y_n)'$, and Σ is the covariance matrix of Y.

The log likelihood function of the original data $X_1, \cdots, X_n$ is

$$l_X(\cdot) = l_Y(\cdot) + (\lambda - 1)\sum_{t=1}^{n} \ln(X_t + c)$$

where *c* is the value of the CONST= option.

For each value of λ, the maximum log likelihood of the original data is obtained from the maximum log likelihood of the transformed data given the maximum likelihood estimate of the autoregressive model.

The maximum log likelihood values are used to compute the Akaike Information Criterion (AIC) and Schwarz's Bayesian Criterion (SBC) for each λ value. The residual mean square error based on the maximum likelihood estimator is also produced. To compute the mean square error, the predicted values from the model are re-transformed to the original scale (Pankratz 1983, pp. 256-258, and Taylor 1986).

After differencing as specified by the DIF= option, the process is assumed to be a stationary autoregressive process. You can check for stationarity of the series with the %DFTEST macro. If the process is not stationary, differencing with the DIF= option is recommended. For a process with moving average terms, a large value for the AR= option may be appropriate.

DFPVALUE Macro

The %DFPVALUE macro computes the significance of the Dickey-Fuller test. The %DFPVALUE macro evaluates the *p*-value for the Dickey-Fuller test statistic τ for the test of H_0: "The time series has a unit root" vs. H_a: "The time series is stationary" using tables published by Dickey (1976) and Dickey, Hasza, and Fuller (1984).

The %DFPVALUE macro can compute *p*-values for tests of a simple unit root with lag 1 or for seasonal unit roots at lags 2, 4, or 12. The %DFPVALUE macro takes into account whether an intercept or deterministic time trend is assumed for the series.

The %DFPVALUE macro is used by the %DFTEST macro described later in this chapter.

Note that the %DFPVALUE macro has been superseded by the PROBDF function described later in this chapter. It remains for compatibility with past releases of SAS/ETS.

Syntax

The %DFPVALUE macro has the following form:

%DFPVALUE *(tau , nobs [, options])*

The first argument, *tau*, specifies the value of the Dickey-Fuller test statistic.

The second argument, *nobs*, specifies the number of observations on which the test statistic is based.

The first two arguments are required. The following options can be used with the %DFPVALUE macro. Options must follow the required arguments and are separated by commas.

DLAG= 1 | 2 | 4 | 12
specifies the lag period of the unit root to be tested. DLAG=1 specifies a 1-period unit root test. DLAG=2 specifies a test for a seasonal unit root with lag 2. DLAG=4 specifies a test for a seasonal unit root with lag 4. DLAG=12 specifies a test for a seasonal unit root with lag 12. The default is DLAG=1.

TREND= 0 | 1 | 2
specifies the degree of deterministic time trend included in the model. TREND=0 specifies no trend and assumes the series has a zero mean. TREND=1 includes an intercept term. TREND=2 specifies both an intercept and a deterministic linear time trend term. The default is TREND=1. TREND=2 is not allowed with DLAG=2, 4, or 12.

Results

The computed *p*-value is returned in the macro variable &DFPVALUE. If the *p*-value is less than 0.01 or larger than 0.99, the macro variable &DFPVALUE is set to 0.01 or 0.99, respectively.

Details

Minimum Observations

The minimum number of observations required by the %DFPVALUE macro depends on the value of the DLAG= option. The minimum observations are as follows:

DLAG=	Min. Obs.
1	9
2	6
4	4
12	12

DFTEST Macro

The %DFTEST macro performs the Dickey-Fuller unit root test. You can use the %DFTEST macro to decide if a time series is stationary and to determine the order of differencing required for the time series analysis of a nonstationary series.

Most time series analysis methods require that the series to be analyzed is stationary. However, many economic time series are nonstationary processes. The usual approach to this problem is to difference the series. A time series which can be made stationary by differencing is said to have a *unit root*. For more information, see the discussion of this issue in the "Getting Started" section on page 366 of Chapter 11, "The ARIMA Procedure."

The Dickey-Fuller test is a method for testing whether a time series has a unit root. The %DFTEST macro tests the hypothesis H_0: "The time series has a unit root" vs. H_a: "The time series is stationary" based on tables provided in Dickey (1976) and Dickey, Hasza, and Fuller (1984). The test can be applied for a simple unit root with lag 1, or for seasonal unit roots at lag 2, 4, or 12.

Note that the %DFTEST macro has been superseded by the PROC ARIMA stationarity tests. See Chapter 11, "The ARIMA Procedure," for details.

Syntax

The %DFTEST macro has the following form:

%DFTEST *(SAS-data-set , variable [, options])*

The first argument, *SAS-data-set*, specifies the name of the SAS data set containing the time series variable to be analyzed.

The second argument, *variable*, specifies the time series variable name to be analyzed.

The first two arguments are required. The following options can be used with the %DFTEST macro. Options must follow the required arguments and are separated by commas.

AR= *n*
specifies the order of autoregressive model fit after any differencing specified by the DIF= and DLAG= options. The default is AR=3.

DIF= *(differencing-list)*
specifies the degrees of differencing to be applied to the series. The differencing list is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the series be differenced once at lag 1 and once at lag 12. For more details, see the "IDENTIFY Statement" section on page 397 in Chapter 11, "The ARIMA Procedure."

If the option DIF=(d_1, $\cdots$, d_k) is specified, the series analyzed is $(1 - B^{d_1})\cdots(1 - B^{d_k})Y_t$, where Y_t is the variable specified,

and B is the backshift operator defined by $BY_t = Y_{t-1}$.

DLAG= 1 | 2 | 4 | 12
specifies the lag to be tested for a unit root. The default is DLAG=1.

OUT= *SAS-data-set*
writes residuals to an output data set.

OUTSTAT= *SAS-data-set*
writes the test statistic, parameter estimates, and other statistics to an output data set.

TREND= 0 | 1 | 2
specifies the degree of deterministic time trend included in the model. TREND=0 includes no deterministic term and assumes the series has a zero mean. TREND=1 includes an intercept term. TREND=2 specifies an intercept and a linear time trend term. The default is TREND=1. TREND=2 is not allowed with DLAG=2, 4, or 12.

Results

The computed *p*-value is returned in the macro variable &DFTEST. If the *p*-value is less than 0.01 or larger than 0.99, the macro variable &DFTEST is set to 0.01 or 0.99, respectively. (The same value is given in the macro variable &DFPVALUE returned by the %DFPVALUE macro, which is used by the %DFTEST macro to compute the *p*-value.)

Results can be stored in SAS data sets with the OUT= and OUTSTAT= options.

Details

Minimum Observations

The minimum number of observations required by the %DFTEST macro depends on the value of the DLAG= option. Let s be the sum of the differencing orders specified by the DIF= option, let t be the value of the TREND= option, and let p be the value of the AR= option. The minimum number of observations required is as follows:

DLAG=	Min. Obs.
1	$1 + p + s + \max(9, p + t + 2)$
2	$2 + p + s + \max(6, p + t + 2)$
4	$4 + p + s + \max(4, p + t + 2)$
12	$12 + p + s + \max(12, p + t + 2)$

Observations are not used if they have missing values for the series or for any lag or difference used in the autoregressive model.

LOGTEST Macro

The %LOGTEST macro tests whether a logarithmic transformation is appropriate for modeling and forecasting a time series. The logarithmic transformation is often used for time series that show exponential growth or variability proportional to the level of the series.

The %LOGTEST macro fits an autoregressive model to a series and fits the same model to the log of the series. Both models are estimated by the maximum likelihood method, and the maximum log likelihood values for both autoregressive models are computed. These log likelihood values are then expressed in terms of the original data and compared.

You can control the order of the autoregressive models. You can also difference the series and the log transformed series before the autoregressive model is fit.

You can print the log likelihood values and related statistics (AIC, SBC, and MSE) for the autoregressive models for the series and the log transformed series. You can also output these statistics to a SAS data set.

Syntax

The %LOGTEST macro has the following form:

%LOGTEST(*SAS-data-set , variable ,***[***options***])**

The first argument, *SAS-data-set*, specifies the name of the SAS data set containing the time series variable to be analyzed. The second argument, *variable*, specifies the time series variable name to be analyzed.

The first two arguments are required. The following options can be used with the %LOGTEST macro. Options must follow the required arguments and are separated by commas.

AR= *n*
specifies the order of the autoregressive model fit to the series and the log transformed series. The default is AR=5.

CONST= *value*
specifies a constant to be added to the series before transformation. Use the CONST= option when some values of the series are 0 or negative. The series analyzed must be greater than the negative of the CONST= value. The default is CONST=0.

DIF= *(differencing-list)*
specifies the degrees of differencing applied to the original and log transformed series before fitting the autoregressive model. The *differencing-list* is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the transformed series be differenced once at lag 1 and once at lag 12. For more details, see the "IDENTIFY Statement" section on page 397 in Chapter 11, "The ARIMA Procedure."

OUT= *SAS-data-set*
writes the results to an output data set. The output data set includes a variable TRANS identifying the transformation (LOG or NONE), the log likelihood value (LOGLIK), residual mean square error (RMSE), Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC) for the log transformed and untransformed cases.

PRINT= YES | NO
specifies whether the results are printed. The default is PRINT=NO. The printed output shows the log likelihood value, residual mean square error, Akaike Information

Criterion (AIC), and Schwarz's Bayesian Criterion (SBC) for the log transformed and untransformed cases.

Results

The result of the test is returned in the macro variable &LOGTEST. The value of the &LOGTEST variable is 'LOG' if the model fit to the log transformed data has a larger log likelihood than the model fit to the untransformed series. The value of the &LOGTEST variable is 'NONE' if the model fit to the untransformed data has a larger log likelihood. The variable &LOGTEST is set to 'ERROR' if the %LOGTEST macro is unable to compute the test due to errors.

Results are printed when the PRINT=YES option is specified. Results are stored in SAS data sets when the OUT= option is specified.

Details

Assume that a time series X_t is a stationary *p*th order autoregressive process with normally distributed white noise innovations. That is,

$$(1 - \Theta(B))(X_t - \mu_{\mathbf{x}}) = \epsilon_t$$

where $\mu_{\mathbf{x}}$ is the mean of X_t.

The log likelihood function of X_t is

$$\begin{aligned} l_1(\cdot) = & - \frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_{\mathbf{xx}}|) - \frac{n}{2}\ln(\sigma_{\mathbf{e}}^2) \\ & - \frac{1}{2\sigma_{\mathbf{e}}^2}(\mathbf{X} - \mathbf{1}\mu_{\mathbf{x}})'\Sigma_{\mathbf{xx}}^{-1}(\mathbf{X} - \mathbf{1}\mu_{\mathbf{x}}) \end{aligned}$$

where *n* is the number of observations, **1** is the *n*-dimensional column vector of 1s, $\sigma_{\mathbf{e}}^2$ is the variance of the white noise, $\mathbf{X} = (X_1, \cdots, X_n)'$, and $\Sigma_{\mathbf{xx}}$ is the covariance matrix of $\mathbf{X}$.

On the other hand, if the log transformed time series $Y_t = \ln(X_t + c)$ is a stationary *p*th order autoregressive process, the log likelihood function of X_t is

$$\begin{aligned} l_0(\cdot) = & - \frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_{\mathbf{yy}}|) - \frac{n}{2}\ln(\sigma_{\mathbf{e}}^2) \\ & - \frac{1}{2\sigma_{\mathbf{e}}^2}(\mathbf{Y} - \mathbf{1}\mu_{\mathbf{y}})'\Sigma_{\mathbf{yy}}^{-1}(\mathbf{Y} - \mathbf{1}\mu_{\mathbf{y}}) - \sum_{t=1}^{n}\ln(X_t + c) \end{aligned}$$

where $\mu_{\mathbf{y}}$ is the mean of Y_t, $\mathbf{Y} = (Y_1, \cdots, Y_n)'$, and $\Sigma_{\mathbf{yy}}$ is the covariance matrix of $\mathbf{Y}$.

The %LOGTEST macro compares the maximum values of $l_1(\cdot)$ and $l_0(\cdot)$ and determines which is larger.

The %LOGTEST macro also computes the Akaike Information Criterion (AIC), Schwarz's Bayesian Criterion (SBC), and residual mean square error based on the maximum likelihood estimator for the autoregressive model. For the mean square error, retransformation of forecasts is based on Pankratz (1983, pp. 256-258).

After differencing as specified by the DIF= option, the process is assumed to be a stationary autoregressive process. You may wish to check for stationarity of the series using the %DFTEST macro. If the process is not stationary, differencing with the DIF= option is recommended. For a process with moving average terms, a large value for the AR= option may be appropriate.

Financial Functions

This section contains the data step financial functions provided with SAS/ETS software in addition to the financial functions in Base SAS software. Refer to *SAS Language Reference: Functions and CALL Routines* for more details on data step functions in general, and for other financial functions not described in this chapter.

The PMT, CUMIPMT, CUMPPMT, PPMT, and IPMT functions are focused on calculations of loans and savings. The EFFRATE and NOMRATE functions perform conversion between effective and nominal interest rates depending on compounding intervals. TIMEVALUE and SAVINGS functions perform time value of money calculations with interest rates that vary over time.

CUMIPMT Function

The CUMIPMT function returns the cumulative interest paid on a loan between *StartPeriod* and *EndPeriod*.

Syntax

CUMIPMT(*Rate, NumberOfPeriods, PrincipalAmount, StartPeriod, EndPeriod, Type* **)**

Rate	specifies the interest rate per payment period. The argument *Rate* is required.
NumberOfPeriods	specifies the number of payment periods. The argument *NumberOfPeriods* is required. It needs to have a positive integer value.
PrincipalAmount	specifies the principal amount of the loan. The argument *PrincipalAmount* is required. Zero is assumed if a missing value is specified.
StartPeriod	specifies the start period for the calculation.
EndPeriod	specifies the end period for the calculation.

Type	specifies whether the payments are at the beginning or end of a period. 0 represents end of period payments and 1 represents beginning of period payments. 0 (end of period payments) is assumed if omitted or a missing value is specified.

Examples

The cumulative interest paid during the second year of the loan on a $125,000 30-year loan with end of period monthly payments and a nominal annual rate is 9% is specified as:

```
TotalInterest  = CUMIPMT (.09/12, 360, 125000, 13, 24, 0);
```

and returns 11,135.23.

The interest paid on the first period of the same loan is specified as:

```
first_period_interest = CUMIPMT (.09/12, 360, 125000, 1, 1, 0);
```

and returns 937.50.

CUMPRINC Function

The CUMPRINC function returns the cumulative principal paid on a loan between *StartPeriod* and *EndPeriod*.

Syntax

CUMPRINC(*Rate, NumberOfPeriods, PrincipalAmount, StartPeriod, EndPeriod, Type* **)**

Rate	specifies the interest rate per payment period. The argument *Rate* is required.
NumberOfPeriods	specifies the number of payment periods. The argument *NumberOfPeriods* is required. It needs to have a positive integer value.
PrincipalAmount	specifies the principal amount of the loan. The argument *PrincipalAmount* is required. Zero is assumed if a missing value is specified.
StartPeriod	specifies the start period for the calculation.
EndPeriod	specifies the end period for the calculation.
Type	specifies whether the payments are at the beginning or end of a period. ‘0’ represents end of period payments and ‘1’ represents beginning of period payments. ‘0’ (end of period payments) is assumed if omitted or a missing value is specified.

Examples

The cumulative principal paid during the second year of a $125,000 30-year loan with end of period monthly payments and a nominal annual rate of 9% is specified as:

```
PrincipalYear2  = CUMRINC (0.09/12, 360, 125000, 12, 24, 0);
```

and returns 934.107.

The principal paid on the second year of the same loan with beginning of period payments is specified as:

```
PrincipalYear2b = CUMPRINC (0.09/12, 360, 125000, 12, 24, 1);
```

and returns 927.153.

EFFRATE Function

The EFFRATE function returns the effective annual interest rate. EFFRATE computes the effective annual interest rate corresponding to a nominal annual interest rate.

Syntax

EFFRATE(*CompoundingInterval, Rate* **)**

CompoundingInterval	is a SAS interval. This is how often *Rate* compounds.
Rate	is a numeric. This is a nominal annual interest rate (expressed as a percentage) that is compounded each *CompoundingInterval*.

Details

- The values for rates must be at least -99.
- Consider a nominal interest *Rate* and a compounding *CompoundingInterval*. If *CompoundingInterval* is 'CONTINUOUS,' the value returned by EFFRATE(*CompoundingInterval, Rate*) equals

  $$e^{Rate/100} - 1$$

  If *CompoundingInterval* is not 'CONTINUOUS,' and *m* *CompoundingInterval*s occur in a year, the value returned by EFFRATE(*CompoundingInterval, Rate*) equals

  $$\left(1 + \frac{Rate}{100\ m}\right)^m - 1$$

- Valid values for *CompoundingInterval* are: 'CONTINUOUS,' 'DAY,' 'SEMIMONTH,' 'MONTH,' 'QUARTER,' 'SEMIYEAR,' and 'YEAR.'
- If *Interval* is 'DAY,' then $m = 365$.

Examples

Suppose a nominal rate is 10%. The corresponding effective rate when interest is compounded monthly can be expressed as

```
effective_rate1 = EFFRATE ("MONTH" 10);
```

Again, suppose a nominal rate is 10%. The corresponding effective rate when interest is compounded quarterly can be expressed as

```
effective_rate2 = EFFRATE ("QUARTER", 10);
```

IPMT Function

The IPMT function returns the interest payment for a given period for a constant payment loan or the periodic saving for a future balance.

Syntax

IPMT(*Rate, Period, NumberOfPeriods, PrincipalAmount, FutureAmount, Type* **)**

Rate	specifies the interest rate per payment period. The argument *Rate* is required.
Period	specifies the payment period for which the interest payment is computed. The argument *Period* is required. *Period* needs to have a positive integer value less than or equal to the *NumberofPeriods*.
NumberOfPeriods	specifies the number of payment periods. The argument *NumberOfPeriods* is required. It needs to have a positive integer value.
PrincipalAmount	specifies the principal amount of the loan. The argument *PrincipalAmount* is required. Zero is assumed if a missing value is specified.
FutureAmount	specifies the future amount, either outstanding balance after *NumberOfPeriods* in case of a loan or the future balance of periodic savings. Zero is assumed if omitted or a missing value is specified.
Type	specifies whether the payments are at the beginning or end of a period. ‘0’ represents end of period payments and ‘1’ represents beginning of period payments. ‘0’ (end of period payments) is assumed if omitted or a missing value is specified.

Examples

The interest payment on the first periodic payment for a $8,000 loan where the nominal annual rate is 10% and there are 36 end of period monthly payments is specified as:

```
    InterestPaid1 = IPMT(0.1/12, 1, 36, 8000);
```

and returns 66.67.

If the same loan has beginning of period payments then:

```
   InterestPaid2 =IPMT(.1/12, 1, 36, 8000, 0, 1);
```

and returns 0.0.

```
   InterestPaid3 =IPMT(.1, 3, 3, 8000);
```

returns 292.447.

```
  InterestPaid4 = IPMT(0.09/12, 359, 360,125000,0, 1);
```

returns 7.4314473.

NOMRATE Function

The NOMRATE function returns the nominal annual interest rate. NOMRATE computes the nominal annual interest rate corresponding to an effective annual interest rate.

Syntax

NOMRATE(*Interval*, *Rate* **)**

Interval	is a SAS interval. This is how often the returned value is compounded.
Rate	is a numeric. This is an effective annual interest rate (expressed as a percentage) that is compounded each *Interval*.

Details

- The values for rates must be at least -99.
- Consider an effective interest *Rate* and a compounding *Interval*. If *CompoundingInterval* is “CONTINUOUS”, the value returned by NOMRATE(*Interval, Rate*) equals

  $$\log_e\left(1+\frac{Rate}{100}\right)$$

  If *CompoundingInterval* is not “CONTINUOUS” and *m Interval*s occur in a year, the value returned by NOMRATE(*Interval, Rate*) equals

  $$m\left(\left(1+\frac{Rate}{100}\right)^{\frac{1}{m}}-1\right)$$

- Valid values for *CompoundingInterval* are: “CONTINUOUS”, “DAY”, “SEMIMONTH”, “MONTH”, “QUARTER”, “SEMIYEAR”, and “YEAR”.
- If *Interval* is “DAY”, then $m = 365$.

Examples

Suppose an effective rate is 10% when compounding monthly. The corresponding nominal rate can be expressed as

```
effective_rate1 = NOMRATE ("MONTH", 10);
```

Suppose an effective rate is 10% when compounding quarterly. The corresponding nominal rate can be expressed as

```
effective_rate2 = NOMRATE ("QUARTER", 10);
```

PMT Function

The PMT function returns the periodic payment for a constant payment loan or the periodic saving for a future balance.

Syntax

PMT(*Rate, NumberOfPeriods, PrincipalAmount, FutureAmount, Type* **)**

Rate	specifies the interest rate per payment period. The argumentRate is required.
NumberOfPeriods	specifies the number of payment periods. The *NumberOfPeriods* is required. NumerofPeriods needs to have a positive integer value.

PrincipalAmount	specifies the principal amount of the loan. The argument *PrincipalAmount* is a required argument. Zero is assumed if a missing value is specified.
FutureAmount	specifies the future amount, either outstanding balance after *NumberOfPeriods* in case of a loan or the future balance of periodic savings. Zero is assumed if omitted or a missing value is specified.
Type	specifies whether the payments are at the beginning or end of a period. ‘0’ represents end of period payments and 1 represents beginning of period payments. ‘0’ (end of period payments) is assumed if omitted or a missing value is specified.

Examples

The monthly payment for a $10,000 loan with a nominal annual rate of 8% and 10 end of month payments is specified as:

```
    Payment1 = PMT (0.08/12, 10, 10000, 0, 0);
```

or

```
   Payment1 = PMT (0.08/12, 10, 10000);
```

and returns 1037.03.

If the same loan has beginning of period payments then:

```
   Payment2 = PMT  (0.08/12, 10, 10000, 0, 1);
```

and returns 1030.16.

If you loan $5000 to be paid back to you in 5 monthly payments earning a 12% annual nominal rate, it is specified as:

```
   Payment3= PMT (.01, 5, -5000);
```

and returns -1030.20

The monthly periodic savings over 18 years earning a 6% annual nominal interest rate which would accumulate $50,000 at the end of the 18 years is specified as:

```
   payment3 = PMT (0.06/12, 216, 0, 50000, 0);
```

and returns 129.081.

PPMT Function

The PPMT function returns the principal payment for a given period for a constant payment loan or the periodic saving for a future balance.

Syntax

PPMT(*Rate, Period, NumberOfPeriods, PrincipalAmount, FutureAmount, Type* **)**

Rate	specifies the interest rate per payment period. The argument *Rate* is required.
Period	specifies the payment periods for which the principal payment is computed. The argument *Period* is required. *Period* needs to have a positive integer value less than or equal to the *NumberofPeriods*.
NumberOfPeriods	specifies the number of payment periods. The argument *NumberOfPeriods* is required. It needs to have a positive integer value.
PrincipalAmount	specifies the principal amount of the loan. The argument *PrincipalAmount* is a required argument. Zero is assumed if a missing value is specified.
FutureAmount	specifies the future amount, either outstanding balance after *NumberOfPeriods* in case of a loan or the future balance of periodic savings. '0' is assumed if omitted or a missing value is specified.
Type	specifies whether the payments are at the beginning or end of a period. '0' represents end of period payments and 1 represents beginning of period payments. '0' (end of period payments) is assumed if omitted or a missing value is specified.

Examples

The principal payment amount of the first monthly periodic payment for a 2 year $2,000 loan with a nominal annual rate of 10% is specified as:

```
PrincipalPayment = PPMT(.1/12, 1, 24, 2000);
```

and returns 75.62.

A 3-year $20,000 loan with beginning of month payments is specified as:

```
PrincipalPayment2  =PPMT(.1/12, 1, 36, 20000, 0, 1);
```

and returns $640.10 as the principal paid with the first payment.

An end-of-month payment loan with an outstanding balance of $5,000 at the end of 3 years is specified as:

```
PrincipalPayment3 =PPMT(.1/12, 1, 36, 20000, 5000, 0);
```

and returns $389.914 as the principal paid with the first payment.

SAVINGS Function

The SAVINGS function returns the balance of a periodic savings using variable interest rates.

Syntax

SAVINGS(*BaseDate, InitialDepositDate, DepositAmount, DepositNumber, DepositInterval, CompoundingInterval, Date-1, Rate-1* **[** , *Date-2, Rate-2,* ...**]** *)*

BaseDate	is a SAS date. The returned value is the balance of the savings at *BaseDate*.
InitialDepositDate	is a SAS date. This is the date of the first deposit. Subsequent deposits are at the beginning of subsequent deposit intervals.
DepositAmount	is a numeric. All deposits are assumed constant. This is the value of each deposit.
DepositNumber	is a positive integer. This is the number of deposits.
DepositInterval	is a SAS interval. This is the frequency at which you make deposits.
CompoundingInterval	is a SAS interval. This is the compounding interval.
Date-i	is a SAS date. Each date is paired with a rate. The date *Date-i* is the time that *Rate-i* takes effect.
Rate-i	is a numeric percentage. Each rate is paired with a date. The rate *Rate-i* is the interest rate that starts on *Date-i*.

Details

- The values for rates must be between -99 and 120.
- *DepositInterval* cannot be ‘CONTINUOUS’
- The list of date-rate pairs does not need to be given in chronological order.
- When multiple rate changes occur on a single date, SAS applies only the final rate listed for that date.
- Simple interest is applied for partial periods.
- There must be a valid date-rate pair whose date is at or prior to both the *InitialDepositDate* and *BaseDate*.

Examples

Suppose you deposit $300 monthly for two years into an account that compounds quarterly at an annual rate of 4%. The balance of the account after five years can be expressed as

```
amount_base1 = SAVINGS ("01jan2005"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4.00);
```

Suppose the interest rate increases by a quarter-point each year. Then the balance of the account could be expressed as

```
amount_base2 = SAVINGS ("01jan2005"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4.00,
                        "01jan2001"d, 4.25, "01jan2002"d, 4.50,
                        "01jan2003"d, 4.75, "01jan2004"d, 5.00);
```

If you want to know the balance after one year of deposits, the following statement sets amount_base3 to the desired balance.

```
amount_base3 = SAVINGS ("01jan2001"d, "01jan2000"d, 300, 24,
                        "MONTH", "QUARTER", "01jan2000"d, 4);
```

Recall that SAS ignores deposits after the base date, so the deposits after the *ReferenceDate* do not affect the returned value.

TIMEVALUE Function

The TIMEVALUE function returns the equivalent of a reference amount at a base date using variable interest rates. TIMEVALUE computes the time-value equivalent of a date-amount pair at a specified date.

Syntax

TIMEVALUE(*BaseDate, ReferenceDate, ReferenceAmount, CompoundingInterval, Date-1, Rate-1* **[** *, Date-2, Rate-2, ...* **]** *)*

BaseDate	is a SAS date. The returned value is the time value of *ReferenceAmount* at *BaseDate*.
ReferenceDate	is a SAS date. This is the date of *ReferenceAmount*.
ReferenceAmount	is a numeric. This is the amount at *ReferenceDate*.
CompoundingInterval	is a SAS interval. This is the compounding interval.
Date-i	is a SAS date. Each date is paired with a rate. The date *Date-i* is the time that *Rate-i* takes effect.
Rate-i	is a numeric percentage. Each rate is paired with a date. The rate *Rate-i* is the interest rate that starts on *Date-i*.

Details

- The values for rates must be between -99 and 120.
- The list of date-rate pairs does not need to be sorted by date.
- When multiple rate changes occur on a single date, SAS applies only the final rate listed for that date.
- Simple interest is applied for partial periods.
- There must be a valid date-rate pair whose date is at or prior to both the *ReferenceDate* and *BaseDate*.

Examples

You can express the accumulated value of an investment of $1,000 at a nominal interest rate of 10% compounded monthly for one year as

```
amount_base1 = TIMEVALUE ("01jan2001"d, "01jan2000"d, 1000,
                          "MONTH", "01jan2000"d, 10);
```

If the interest rate jumps to 20% halfway through the year, the resulting calculation would be

```
amount_base2 = TIMEVALUE ("01jan2001"d, "01jan2000"d, 1000,
                          "MONTH",
                          "01jan2000"d, 10, "01jul2000"d, 20);
```

Recall that the date-rate pairs do not need to be sorted by date. This flexibility allows amount_base2 and amount_base3 to assume the same value.

```
amount_base3 = TIMEVALUE ("01jan2001"d, "01jan2000"d, 1000,
                          "MONTH",
                          "01jul2000"d, 20, "01jan2000"d, 10);
```

Other Functions

PROBDF Function for Dickey-Fuller Tests

The PROBDF function calculates significance probabilities for Dickey-Fuller tests for unit roots in time series. The PROBDF function can be used wherever SAS library functions may be used, including DATA step programs, SCL programs, and PROC MODEL programs.

Syntax

PROBDF(*x*, *n* **[** , *d* **[** , *type* **]])**

x	is the test statistic.
n	is the sample size. The minimum value of *n* allowed depends on the value specified for the second argument *d*. For *d* in the set (1,2,4,6,12), *n* must be an integer greater than or equal to $\max(2d, 5)$; for other values of *d* the minimum value of *n* is 24.
d	is an optional integer giving the degree of the unit root tested for. Specify $d = 1$ for tests of a simple unit root $(1 - \mathrm{B})$. Specify *d* equal to the seasonal cycle length for tests for a seasonal unit root $(1 - \mathrm{B}^{\mathrm{d}})$. The default value of *d* is 1; that is, a test for a simple unit root $(1 - \mathrm{B})$ is assumed if *d* is not specified. The maximum value of *d* allowed is 12.
type	is an optional character argument that specifies the type of test statistic used. The values of *type* are

SZM	studentized test statistic for the zero mean (no intercept) case
RZM	regression test statistic for the zero mean (no intercept) case
SSM	studentized test statistic for the single mean (intercept) case
RSM	regression test statistic for the single mean (intercept) case
STR	studentized test statistic for the deterministic time trend case
RTR	regression test statistic for the deterministic time trend case

The values STR and RTR are allowed only when $d = 1$. The default value of *type* is SZM.

Details

Theoretical Background

When a time series has a unit root, the series is nonstationary and the ordinary least squares (OLS) estimator is not normally distributed. Dickey (1976) and Dickey and Fuller (1979) studied the limiting distribution of the OLS estimator of autoregressive models for time series with a simple unit root. Dickey, Hasza, and Fuller (1984) obtained the limiting distribution for time series with seasonal unit roots.

Consider the (*p*+1)th order autoregressive time series

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \cdots + \alpha_{p+1} Y_{t-p-1} + e_t$$

and its characteristic equation

$$m^{p+1} - \alpha_1 m^p - \alpha_2 m^{p-1} - \cdots - \alpha_{p+1} = 0$$

If all the characteristic roots are less than 1 in absolute value, Y_t is stationary. Y_t is nonstationary if there is a unit root. If there is a unit root, the sum of the autoregressive parameters is 1, and, hence, you can test for a unit root by testing whether the sum of the autoregressive parameters is 1 or not. For convenience, the model is parameterized as

$$\nabla Y_t = \delta Y_{t-1} + \theta_1 \nabla Y_{t-1} + \cdots + \theta_p \nabla Y_{t-p} + e_t$$

where $\nabla Y_t = Y_t - Y_{t-1}$ and

$$\delta = \alpha_1 + \cdots + \alpha_{p+1} - 1$$

$$\theta_k = -\alpha_{k+1} - \cdots - \alpha_{p+1}$$

The estimators are obtained by regressing ∇Y_t on $Y_{t-1}, \nabla Y_{t-1}, \cdots, \nabla Y_{t-p}$. The *t* statistic of the ordinary least squares estimator of δ is the test statistic for the unit root test.

If the TREND=1 option is used, the autoregressive model includes a mean term α_0. If TREND=2, the model also includes a time trend term and the model is as follows:

$$\nabla Y_t = \alpha_0 + \gamma t + \delta Y_{t-1} + \theta_1 \nabla Y_{t-1} + \cdots + \theta_p \nabla Y_{t-p} + e_t$$

For testing for a seasonal unit root, consider the multiplicative model

$$(1 - \alpha_d B^d)(1 - \theta_1 B - \cdots - \theta_p B^p) Y_t = e_t$$

Let $\nabla^d Y_t \equiv Y_t - Y_{t-d}$. The test statistic is calculated in the following steps:

1. Regress $\nabla^d Y_t$ on $\nabla^d Y_{t-1} \cdots \nabla^d Y_{t-p}$ to obtain the initial estimators $\hat{\theta}_i$ and compute residuals $\hat{e}_t$. Under the null hypothesis that $\alpha_d = 1$, $\hat{\theta}_i$ are consistent estimators of θ_i.
2. Regress $\hat{e}_t$ on $(1 - \hat{\theta}_1 B - \cdots - \hat{\theta}_p B^p) Y_{t-d}, \nabla^d Y_{t-1}, \cdots, \nabla^d Y_{t-p}$to obtain estimates of $\delta = \alpha_d - 1$ and $\theta_i - \hat{\theta}_i$.

The *t* ratio for the estimate of δ produced by the second step is used as a test statistic for testing for a seasonal unit root. The estimates of θ_i are obtained by adding the estimates of $\theta_i - \hat{\theta}_i$ from the second step to $\hat{\theta}_i$ from the first step. The estimates of $\alpha_d - 1$ and θ_i are saved in the OUTSTAT= data set if the OUTSTAT= option is specified.

The series $(1 - B^d) Y_t$ is assumed to be stationary, where *d* is the value of the DLAG= option.

If the OUTSTAT= option is specified, the OUTSTAT= data set contains estimates $\hat{\delta}, \hat{\theta}_1, \cdots, \hat{\theta}_p$.

If the series is an ARMA process, a large value of the AR= option may be desirable in order to obtain a reliable test statistic. To determine an appropriate value for the AR= option for an ARMA process, refer to Said and Dickey (1984).

Test Statistics

The Dickey-Fuller test is used to test the null hypothesis that the time series exhibits a lag *d* unit root against the alternative of stationarity. The PROBDF function computes the probability of observing a test statistic more extreme than *x* under the assumption that the null hypothesis is true. You should reject the unit root hypothesis when PROBDF returns a small (significant) probability value.

There are several different versions of the Dickey-Fuller test. The PROBDF function supports six versions, as selected by the *type* argument. Specify the *type* value that corresponds to the way that you calculated the test statistic *x*.

The last two characters of the *type* value specify the kind of regression model used to compute the Dickey-Fuller test statistic. The meaning of the last two characters of the *type* value are as follows.

ZM	zero mean or no intercept case. The test statistic *x* is assumed to be computed from the regression model $y_t = \alpha_d y_{t-d} + e_t$
SM	single mean or intercept case. The test statistic *x* is assumed to be computed from the regression model $y_t = \alpha_0 + \alpha_d y_{t-d} + e_t$
TR	intercept and deterministic time trend case. The test statistic *x* is assumed to be computed from the regression model $y_t = \alpha_0 + \gamma t + \alpha_1 y_{t-1} + e_t$

The first character of the *type* value specifies whether the regression test statistic or the studentized test statistic is used. Let $\hat{\alpha}_d$ be the estimated regression coefficient for the *d*th lag of the series, and let $\mathrm{se}_{\hat{\alpha}}$ be the standard error of $\hat{\alpha}_d$. The meaning of the first character of the *type* value is as follows.

R	the regression coefficient-based test statistic. The test statistic is $x = n(\hat{\alpha}_d - 1)$
S	the studentized test statistic. The test statistic is $x = \dfrac{(\hat{\alpha}_d - 1)}{\mathrm{se}_{\hat{\alpha}}}$

Refer to Dickey and Fuller (1979) and Dickey, Hasza, and Fuller (1984) for more information about the Dickey-Fuller test null distribution. The preceding formulas are for the basic Dickey-Fuller test. The PROBDF function can also be used for the augmented Dickey-Fuller test, in which the error term e_t is modeled as an autoregressive process; however, the test statistic is computed somewhat differently for the augmented Dickey-Fuller test. Refer to Dickey, Hasza, and Fuller (1984) and Hamilton (1994) for information about seasonal and nonseasonal augmented Dickey-Fuller tests.

The PROBDF function is calculated from approximating functions fit to empirical quantiles produced by Monte Carlo simulation employing 10^8 replications for each simulation. Separate simulations were performed for selected values of *n* and for $d = 1, 2, 4, 6, 12$.

The maximum error of the PROBDF function is approximately $\pm 10^{-3}$ for *d* in the set (1,2,4,6,12) and may be slightly larger for other *d* values. (Because the number of simulation replications used to produce the PROBDF function is much greater than the 60,000 replications used by Dickey and Fuller (1979) and Dickey, Hasza, and Fuller (1984), the PROBDF function can be expected to produce results that are substantially more accurate than the critical values reported in those papers.)

Examples

Suppose the data set TEST contains 104 observations of the time series variable Y, and you want to test the null hypothesis that there exists a lag 4 seasonal unit root in the Y series. The following statements illustrate how to perform the single-mean Dickey-Fuller regression coefficient test using PROC REG and PROBDF.

```
data test1;
   set test;
   y4 = lag4(y);
run;

proc reg data=test1 outest=alpha;
   model y = y4 / noprint;
run;

data _null_;
   set alpha;
   x = 100 * ( y4 - 1 );
   p = probdf( x, 100, 4, "RSM" );
   put p= pvalue5.3;
run;
```

To perform the augmented Dickey-Fuller test, regress the differences of the series on lagged differences and on the lagged value of the series, and compute the test statistic from the regression coefficient for the lagged series. The following statements illustrate how to perform the single-mean augmented Dickey-Fuller studentized test using PROC REG and PROBDF.

```
data test1;
   set test;
   yl  = lag(y);
   yd  = dif(y);
```

```
      yd1 = lag1(yd); yd2 = lag2(yd);
      yd3 = lag3(yd); yd4 = lag4(yd);
   run;

   proc reg data=test1 outest=alpha covout;
      model yd = yl yd1-yd4 / noprint;
   run;

   data _null_;
      set alpha;
      retain a;
      if _type_ = 'PARMS' then a = yl - 1;
      if _type_ = 'COV' & _NAME_ = 'YL' then do;
         x = a / sqrt(yl);
         p = probdf( x, 99, 1, "SSM" );
         put p= pvalue5.3;
         end;
   run;
```

The %DFTEST macro provides an easier way to perform Dickey-Fuller tests. The following statements perform the same tests as the preceding example.

```
   %dftest( test, y, ar=4 );
   %put p=&dftest;
```

References

Dickey, D. A. (1976), "Estimation and Testing of Nonstationary Time Series," Unpublished Ph.D. Thesis, Iowa State University, Ames.

Dickey, D. A. and Fuller, W. A. (1979), "Distribution of the Estimation for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, 74, 427-431.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984), "Testing for Unit Roots in Seasonal Time Series," *Journal of the American Statistical Association*, 79, 355-367.

Fuller, W. A. (1976), *Introduction to Statistical Time Series*. New York: John Wiley.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton, NJ: Princeton University Press.

Microsoft Excel 2000 Online Help, Redmond, WA: Microsoft Corp.

Pankratz, A. (1983), *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*. New York: John Wiley.

Said, S. E. and Dickey, D. A. (1984), "Testing for Unit Roots in ARMA Models of Unknown Order," *Biometrika*, 71, 599-607.

Taylor, J. M. G. (1986) "The Retransformed Mean After a Fitted Power Transformation," *Journal of the American Statistical Association*, 81, 114-118.

Chapter 5
The SASECRSP Interface Engine

Chapter Contents

Chapter 5 The SASECRSP Interface Engine

Overview

The SASECRSP interface engine enables SAS users to access and process time series data residing in CRSPAccess data files and provides a seamless interface between CRSP and SAS data processing.

The SASECRSP engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the CRSPAccess data files, and how you would like to perform selection on the CRSP set you choose to access. You choose the daily CRSP data by specifying SETID=10, or the monthly CRSP data by specifying SETID=20. You can select which securities you wish to access by specifying PERMNO=*number*, the primary key, or you can select which securities you wish to access by specifying a secondary key, such as PERMCO=*number*, a secondary key, a unique permanent identifier assigned by CRSP to all companies with issues on the CRSP Stock Files, or CUSIP=*number*, a secondary key, assigned to individual securities by the Standard and Poor's CUSIP Service, or HCUSIP=*number*, a secondary key, historical cusip for a security, or TICKER=*character*, a secondary key, ticker symbol for a security, or SICCD=*character*, a secondary key, standard industrial classification code, and you can specify your range of dates for the selected time series by using RANGE=‘*begdt-enddt*’, or specify an input SAS data set named *setname* as input for issues with the INSET=‘*setname*’ option. The SAS Data step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time. Since CRSP and SAS use three different date representations, you can use the engine-provided CRSP date formats, informats, and functions for flexible seamless programming.

SASECRSP for Version 9 supports Windows, Solaris 8 (SUNOS5.8), LINUX and DEC/OSF Digital UNIX.

Getting Started

Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form that is recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASECRSP engine provides a time ID variable called CALDT.

Reading CRSP Data Files

The SASECRSP engine supports reading time series, events, and portfolios from CRSPAccess data files. Only the series specified by the CRSP *setid* is written to the SAS data set. The CRSP environment variable CRSPDB_SASCAL must be defined to allow the SASECRSP engine access to the CRSPAccess database calendars.

Using the SAS DATA Step

If desired, you can store the selected series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once the data is stored in a SAS data set you can use it as you would any other SAS data set.

Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information concerning the contents of your data set by using the CONTENTS procedure. You can create a view of the CRSPAccess data base by using the SQL procedure to create your view using the SASECRSP engine in your *libref*, along with the USING clause.

Using CRSP Date Formats, Informats, and Functions

CRSP has historically used two different methods to represent dates, and SAS has used a third. The SASECRSP engine provides 22 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. See the section "Understanding CRSP Date Formats, Informats, and Functions" on page 170 for details.

Syntax

The SASECRSP engine uses standard engine syntax. Options used by SASECRSP are summarized in the table below.

Description	Statement	Option
specify a CRSPAccess set id where 10 is daily and 20 monthly. This is a required option.	LIBNAME *libref* SASECRSP	SETID=
specify the PERMNO of a security to be kept in the SAS data set.	LIBNAME *libref* SASECRSP	PERMNO=
specify the PERMCO of a security to be kept in the SAS data set,	LIBNAME *libref* SASECRSP	PERMCO=
specify the CUSIP of a security to be kept in the SAS data set,	LIBNAME *libref* SASECRSP	CUSIP=
specify the HCUSIP of a security to be kept in the SAS data set,	LIBNAME *libref* SASECRSP	HCUSIP=
specify the TICKER of a security to be kept in the SAS data set,	LIBNAME *libref* SASECRSP	TICKER=
specify the SICCD of a security to be kept in the SAS data set,	LIBNAME *libref* SASECRSP	SICCD=
specify the range of data in format YYYYMMDD using the *crsp_begdt* and *crsp_enddt*.	LIBNAME *libref* SASECRSP	RANGE=
use a SAS data set named *setname* as input for issues.	LIBNAME *libref* SASECRSP	INSET=

The LIBNAME *libref* SASECRSP Statement

LIBNAME *libref* **SASECRSP** **'***physical name***'** *options;*

The CRSP environment variable CRSPDB_SASCAL must be defined to allow the SASECRSP engine access to the CRSPAccess database calendars. Often your CRSPDB_SASCAL will be pointing to the same path as your *physical name*. You must use a fully qualified pathname. Your *physical name* must end with either a forward slash if you are on a UNIX system, or a backward slash if you are on WINNT.

For a more complete description of SASECRSP set names, key fields, and date fields, see Table 5.1.

The following options can be used in the LIBNAME *libref* SASECRSP statement:

SETID=*crsp_setidnumber*

specifies the CRSP set ID number where 10 is the daily data set ID and 20 is the monthly data set ID number. Two possible values for *crsp_setidnumber* are 10 or 20. The SETID limits the frequency selection of time series that are included in the SAS data set. SETID is a required option for the SASECRSP engine. Depending on your host system, both should end in either a forward slash (UNIX) or a backward slash (WINNT). For more details about the CRSPAccess *crsp_setidnumber*, refer to

“Using Multiple Products” in the *CRSPAccess Database Format Installation Guide*. As an example, to access monthly data, you would use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
   SETID=20;
```

PERMNO=*crsp_permnumber*
By default, the SASECRSP engine reads all PERMNOs in the CRSPAccess database that you name in your SASECRSP *libref*. You can limit the time series read from the CRSP database by specifying the PERMNO= option on your LIBNAME statement. From a performance standpoint, the PERMNO= option does random access and reads only the data for the PERMNOs listed. You can also subset using random access based on the secondary keys, PERMCO= option to read selected securities based on the PERMCOs listed, CUSIP= option to read selected securities based on the the CUSIPs listed, HCUSIP= option to read selected securities based on the HCUSIPs listed, SICCD= option to read selected securities based on the SICCDs listed, and TICKER= option to read selected securities based on the TICKERs listed. There is no limit to the number of *crsp_permnumber* options that you can use. As an example, to access monthly data for Microsoft Corporation and for International Business Machine Corporation, you could use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
   SETID=20
   PERMNO=10107
   PERMNO=12490;
```

In like manner, secondary keys PERMCO, CUSIP, HCUSIP, SICCD and TICKER may be used instead of PERMNO= option.

PERMCO=*crsp_permcompany*
There is no limit to the number of *crsp_permcompany* options that you can use.

CUSIP=*crsp_cusip*
There is no limit to the number of *crsp_cusip* options that you can use.

HCUSIP=*crsp_hcusip*
There is no limit to the number of *crsp_hcusip* options that you can use.

TICKER=*crsp_ticker*
There is no limit to the number of *crsp_ticker* options that you can use.

SICCD=*crsp_siccd*
There is no limit to the number of *crsp_siccd* options that you can use.

RANGE=*'crsp_begdt-crsp_enddt'*
To limit the time range of data read from the CRSPAccess database, specify the RANGE= option in your SASECRSP *libref*, where *crsp_begdt* is the beginning date in YYYYMMDD format and *crsp_enddt* is the ending date of the range in YYYYMMDD format. From a performance standpoint, the engine reads all the data for a company and then restricts the data range before passing it back.

As an example, to access monthly data for Microsoft Corporation and for International Business Machine Corporation for the first quarter of 2000, you could use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
   SETID=20
   PERMNO=10107
   PERMNO=12490
   RANGE='19990101-19990331';
```

INSET=*'setname[,keyfieldname][,keyfieldtype][,date1field][,date2field]'*

When you specify a SAS data set named *setname* as input for issues, the SASECRSP engine assumes a default PERMNO field containing selected CRSP PERMNOs is present in the data set. If optional parameters are included, the first one is the CRSPAccess supported key name, and the second is the key type. Version 9 supports the PERMNO key field on the INSET option, the PERMCO key field on the INSET option, the CUSIP key field on the INSET option, the HCUSIP key field on the INSET option, the SICCD key field on the INSET option, or the TICKER key field on the INSET option. The third optional parameter is the beginning date or event date, and the fourth is the ending date in YYYYMMDD format. For more details about the CRSPAccess keyfields and keyfieldtypes, refer to "Stock and Indices Data Structures" in the *CRSP Data Description Guide*.

The following example might be used to extract the data for a portfolio of four companies to access monthly data where the range of dates is left open:

```
data testin1;
   permno = 10107; output;
   permno = 12490; output;
   permno = 14322; output;
   permno = 25788; output;
run;

LIBNAME mstk sasecrsp 'physical-name'
   SETID=20
   INSET='testin1';

data a;
   set mstk.prc;
run;

proc print data=a;
run;
```

Suppose you want to restrict the dates for each PERMNO specifically. The example below shows how to specify the INSET= option with the *'setname[,keyfieldname][,keyfieldtype][,date1field][,date2field]'* parameters:

```
data testin2;
   permno = 10107; date1 = 19990101; date2 = 19991231; output;
   permno = 12490; date1 = 19970101; date2 = 19971231; output;
   permno = 14322; date1 = 19990901; date2 = 19991231; output;
   permno = 25788; date1 = 19950101; date2 = 19950331; output;
run;

LIBNAME mstk2 sasecrsp 'physical-name'
   SETID=20
   INSET='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
   set mstk2.prc;
run;

proc print data=b;
run;
```

Details

The SAS Output Data Set

You can use the SAS DATA step to write the selected CRSP data to a SAS data set. This enables the user to easily analyze the data using SAS. The name of the output data set is specified by you on the DATA statement. This causes the engine supervisor to create a SAS data set using the specified name in either the SAS WORK library or, if specified, the USER library.

The contents of the SAS data set include the DATE of each observation, the series names of each series read from the CRSPAccess database, event variables, and the label or description of each series/event.

Available Data Sets

Table 5.1 shows the available data sets provided by the SASECRSP interface. Missing values are represented as '.' in the SAS data set. You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment by selecting the SASECRSP libref in the LIBNAME window that you have previously used in your libname statement. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can view your SAS output observations by double clicking on the desired output data set libref in the libname window of the SAS windowing environment. You can use PROC SQL along with the SASECRSP engine to create a view of your SAS data set.

Table 5.1. Data Sets Available

Dataset	Fields	Label	Type
STKHEAD	PERMNO	PERMNO	Numeric
Header Identification	PERMCO	PERMCO	Numeric
and Summary Data	COMPNO	Nasdaq Company Number	Numeric
	ISSUNO	Nasdaq Issue Number	Numeric
	HEXCD	Exchange Code Header	Numeric
	HSHRCD	Share Code Header	Numeric
	HSICCD	Standard Industrial Classification Code	Numeric
	BEGDT	Begin of Stock Data	Numeric
	ENDDT	End of Stock Data	Numeric
	DLSTCD	Delisting Code Header	Numeric
	HCUSIP	CUSIP Header	Character
	HTICK	Ticker Symbol Header	Character
	HCOMNAM	Company Name Header	Character
NAMES	PERMNO	PERMNO	Numeric
Name History Array	NAMEDT	Names Date	Numeric
	NAMEENDDT	Names Ending Date	Numeric
	SHRCD	Share Code	Numeric
	EXCHCD	Exchange Code	Numeric
	SICCD	Standard Industrial Classification Code	Numeric
	NCUSIP	CUSIP	Numeric
	TICKER	Ticker Symbol	Character
	COMNAM	Company Name	Character
	SHRCLS	Share Class	Numeric
DISTS	PERMNO	PERMNO	Numeric
Distribution Event	DISTCD	Distribution Code	Numeric
Array	DIVAMT	Dividend Cash Amount	Numeric
	FACPR	Factor to Adjust Price	Numeric
	FACSHR	Factor to Adjust Share	Numeric
	DCLRDT	Distribution Declaration Date	Numeric
	EXDT	Ex-Distribution Date	Numeric
	RCRDDT	Record Date	Numeric
	PAYDT	Payment Date	Numeric
	ACPERM	Acquiring PERMNO	Numeric
	ACCOMP	Acquiring PERMCO	Numeric
SHARES	PERMNO	PERMNO	Numeric
Shares Outstanding	SHROUT	Shares Outstanding	Numeric
Observation Array	SHRSDT	Shares Outstanding Observation Date	Numeric Numeric
	SHRENDDT	Shares Outstanding Observation End Date	Numeric
	SHRFLG	Shares Outstanding Observation Flag	Numeric
DELIST	PERMNO	PERMNO	Numeric

Table 5.1. (continued)

Dataset	Fields	Label	Type
Delisting History Array	DLSTDT	Delisting Date	Numeric
	DLSTCD	Delisting Code	Numeric
	NWPERM	New PERMNO	Numeric
	NWCOMP	New PERMCO	Numeric
	NEXTDT	Delisting Next Price Date	Numeric
	DLAMT	Delisting Amount	Numeric
	DLRETX	Delisting Return Without Dividends	Numeric
	DLPRC	Delisting Price	Numeric
	DLPDT	Delisting Amount Date	Numeric
	DLRET	Delisting Return	Numeric
NASDIN Nasdaq Information Array	PERMNO	PERMNO	Numeric
	TRTSCD	Nasdaq Traits Code	Numeric
	TRTSDT	Nasdaq Traits Date	Numeric
	TRTSENDDT	Nasdaq Traits End Date	Numeric
	NMSIND	Nasdaq National Market Indicator	Numeric
	MMCNT	Market Maker Count	Numeric
	NSDINX	Nasd Index Code	Numeric
PRC Price or Bid/Ask Average Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	PRC	Price or Bid/Ask Aver	Numeric
RET Returns Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	RET	Returns	Numeric
ASKHI Ask or High Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	ASKHI	Ask or High	Numeric
BIDLO Bid or Low Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	BIDLO	Bid or Low	Numeric
BID Bid Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	BID	Bid	Numeric
ASK Ask Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	ASK	Ask	Numeric
RETX Returns without Dividends Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	RETX	Returns w/o Dividends	Numeric
SPREAD Spread Between Bid and Ask Time Series	PERMNO	PERMNO	Numeric
	CALDT	Calendar Trading Date	Numeric
	SPREAD	Spread Between Bid Ask	Numeric
ALTPRC	PERMNO	PERMNO	Numeric

Table 5.1. (continued)

Dataset	Fields	Label	Type
Price Alternate Time Series	CALDT ALTPRC	Calendar Trading Date Price Alternate	Numeric Numeric
VOL Volume Time Series	PERMNO CALDT VOL	PERMNO Calendar Trading Date Volume	Numeric Numeric Numeric
NUMTRD Number of Trades Time Series	PERMNO CALDT NUMTRD	PERMNO Calendar Trading Date Number of Trades	Numeric Numeric Numeric
ALTPRCDT Alternate Price Date Time Series	PERMNO CALDT ALTPRCDT	PERMNO Calendar Trading Date Alternate Price Date	Numeric Numeric Numeric
PORT1 Portfolio Data for Portfolio Type 1	PERMNO CALDT PORT1 STAT1	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 1 Portfolio Statistic for Portfolio Type 1	Numeric Numeric Numeric Numeric Numeric Numeric
PORT2 Portfolio Data for Portfolio Type 2	PERMNO CALDT PORT2 STAT2	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 2 Portfolio Statistic for Portfolio Type 2	Numeric Numeric Numeric Numeric Numeric Numeric
PORT3 Portfolio Data for Portfolio Type 3	PERMNO CALDT PORT3 STAT3	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 3 Portfolio Statistic for Portfolio Type 3	Numeric Numeric Numeric Numeric Numeric Numeric
PORT4 Portfolio Data for Portfolio Type 4	PERMNO CALDT PORT4 STAT4	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 4 Portfolio Statistic for Portfolio Type 4	Numeric Numeric Numeric Numeric Numeric Numeric
PORT5 Portfolio Data for Portfolio Type 5	PERMNO CALDT PORT5 STAT5	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 5 Portfolio Statistic for Portfolio Type 5	Numeric Numeric Numeric Numeric Numeric Numeric
PORT6 Portfolio Data for Portfolio Type 6	PERMNO CALDT PORT6	PERMNO Calendar Trading Date Portfolio Assignment for Portfolio Type 6	Numeric Numeric Numeric Numeric

Table 5.1. (continued)

Dataset	Fields	Label	Type
	STAT6	Portfolio Statistic	Numeric
		for Portfolio Type 6	Numeric
PORT7	PERMNO	PERMNO	Numeric
Portfolio Data for	CALDT	Calendar Trading Date	Numeric
Portfolio Type 7	PORT7	Portfolio Assignment	Numeric
		for Portfolio Type 7	Numeric
	STAT7	Portfolio Statistic	Numeric
		for Portfolio Type 7	Numeric
PORT8	PERMNO	PERMNO	Numeric
Portfolio Data for	CALDT	Calendar Trading Date	Numeric
Portfolio Type 8	PORT8	Portfolio Assignment	Numeric
		for Portfolio Type 8	Numeric
	STAT8	Portfolio Statistic	Numeric
		for Portfolio Type 8	Numeric
PORT9	PERMNO	PERMNO	Numeric
Portfolio Data for	CALDT	Calendar Trading Date	Numeric
Portfolio Type 9	PORT9	Portfolio Assignment	Numeric
		for Portfolio Type 9	Numeric
	STAT9	Portfolio Statistic	Numeric
		for Portfolio Type 9	Numeric

Understanding CRSP Date Formats, Informats, and Functions

CRSP has historically used two different methods to represent dates, while SAS has used a third. The three formats are SAS Dates, CRSP Dates, and Integer Dates. The SASECRSP engine provides 22 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. A SASECRSP libname assign must be active to use these date access methods. See Example 5.5, "Converting Dates Using the CRSP Date Functions."

SAS dates are internally stored as the number of days since January 1, 1960. The SAS method is an industry standard and provides a great deal of flexibility, including a wide variety of informats, formats, and functions.

CRSP dates are designed to ease time series storage and access. Internally the dates are stored as an offset into an array of trading days. Note that there are five different CRSP trading day calendars: Annual, Quarterly, Monthly, Weekly, and Daily. The CRSP method provides fewer missing values and makes trading period calculations very easy. However, there are also many valid calendar dates that are not available in the CRSP trading calendars, and care must be taken when using other dates.

Integer dates are a way to represent dates that are platform independent and maintain the correct sort order. However, the distance between dates is not maintained.

The best way to illustrate these formats is with some sample data. The table below only shows CRSP Daily and Monthly dates.

Table 5.2. Date Representations for Daily and Monthly Data

Date	SAS Date	CRSP Date (Daily)	CRSP Date (Monthly)	Integer Date
July 31, 1962	942	21	440	19620731
August 31, 1962	973	44	441	19620831
December 30, 1998	14,243	9190	NA*	19981230
December 31, 1998	14,244	9191	877	19981231

* Not available if an exact match is requested.

Having an understanding of the internal differences in representing SAS dates, CRSP dates, and CRSP integer dates will help you use the SASECRSP formats, informats, and functions effectively. Always keep in mind the frequency of the CRSP calendar that you are accessing when you specify a CRSP date.

The CRSP Date Formats

There are two types of formats for CRSP dates, and five frequencies are available for each of the two types. The two types are exact dates (CRSPDT*) and range dates (CRSPDR*), where * can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The ten formats are: CRSPDTA, CRSPDTQ, CRSPDTM, CRSPDTW, CRSPDTD, CRSPDRA, CRSPDRQ, CRSPDRM, CRSPDRW, and CRSPDRD.

Here are some samples using the monthly and daily calendar as examples. The Annual (CRSPDTA and CRSPDRA), Quarterly (CRSPDTQ and CRSPDRQ), and the Weekly (CRSPDTW and CRSPDRW) work analogously.

Table 5.3. Sample CRSPDT Formats for Daily and Monthly Data

Date	CRSP Date Daily , Monthly	CRSPDTD8. Daily Date	CRSPDRD8. Daily Range	CRSPDTM8. Monthly Date	CRSPDRM8. Monthly Range
July 31,1962	21 , 440	19620731	19620731 +	19620731	19620630-19620731
August31,1962	44 , 441	19620831	19620831 +	19620831	19620801-19620831
December30, 1998	9190 , NA *	19981230	19981230 +	NA*	NA*
December31, 1998	9191 , 877	19981231	19981231 +	19981231	19981201-19981231

+ Daily ranges will look similar to Monthly Ranges if they are Mondays or immediately following a trading holiday.
* When working with exact matches, no CRSP monthly date exists for December 30, 1998.

The @CRSP Date Informats

There are three types of informats for CRSP dates, and five frequencies are available for each of the three types. The three types are exact (@CRSPDT*), range (@CRSPDR*), and backward (@CRSPDB*) dates, where * can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The fifteen formats are: @CRSPDTA, @CRSPDTQ, @CRSPDTM, @CRSPDTW, @CRSPDTD, @CRSPDRA, @CRSPDRQ, @CRSPDRM, @CRSPDRW, @CRSPDRD, @CRSPDBA, @CRSPDBQ, @CRSPDBM, @CRSPDBW, and @CRSPDBD.

The five CRSPDT* informats find exact matches only. The five CRSPDR* informats look for an exact match, and if it is not found it goes forward, matching the CRSPDR* formats. The five CRSPDB* informats look for an exact match, and if it is not found it goes backward. Here is a sample using only the CRSP monthly calendar as an example, but the daily, weekly, quarterly, and annual frequencies work analogously.

Table 5.4. Sample @CRSP Date Informats Using Monthly Data

Input Date (Integer Date)	CRSP Date CRSPDTM	CRSP Date CRSPDRM	CRSP Date CRSPDBM	CRSPDTM8. Monthly Date	CRSPDRM8. Monthly Range
19620731	440	440	440	19620731	19620630-19620731
19620815	.(missing)	441	440	See below+	See below*
19620831	441	441	441	19620831	19620801-19620831

+ If missing, then missing. If 441, then 19620831. If 440, then 19620731.
* If missing, then missing. If 441, then 19620801-19620831. If 440, then 19620630-19620731.

The CRSP Date Functions

There are 22 date functions provided with the SASECRSP engine. These functions are used internally by the engine, but also are available to the end users. There are six groups of functions. The first four have five functions each, one for each CRSP calendar frequency.

Table 5.5. CRSP Date Functions

Function Group	Function Name	Argument One	Argument Two	Return Value
CRSP dates to Integer dates for December 31, 1998				
Annual	crspdcia	74	None	19981231
Quarterly	crspdciq	293	None	19981231
Monthly	crspdcim	877	None	19981231
Weekly	crspdciw	1905	None	19981231
Daily	crspdcid	9191	None	19981231
CRSP dates to SAS dates for December 31, 1998				
Annual	crspdcsa	74	None	14,244
Quarterly	crspdcsq	293	None	14,244
Monthly	crspdcsm	877	None	14,244
Weekly	crspdcsw	1905	None	14,244
Daily	crspdcsd	9191	None	14,244
Integer dates to CRSP dates exact is illustrated, but can be forward or backward				
Annual	crspdica	19981231	0	74

Table 5.5. (continued)

Function Group	Function Name	Argument One	Argument Two	Return Value
Quarterly	crspdicq	19981231	0	293
Monthly	crspdicm	19981231	0	877
Weekly	crspdicw	19981231	0	1905
Daily	crspdicd	19981231	0	9191
SAS dates to CRSP dates exact is illustrated, but can be forward or backward				
Annual	crspdsca	14,244	0	74
Quarterly	crspdscq	14,244	0	293
Monthly	crspdscm	14,244	0	877
Weekly	crspdscw	14,244	0	1905
Daily	crspdscd	14,244	0	9191
Integer dates to SAS dates for December 31, 1998				
Integer to SAS	crspdi2s	19981231	None	14,244
SAS dates to Integer dates for December 31, 1998				
SAS to Integer	crspds2i	14,244	None	19981231

Examples

Example 5.1. Extracting a List of PERMNOs Using a RANGE

This example specifies a list of PERMNOs that are desired from the monthly set of CRSPAccess data. Suppose you want the range of data starting January 1, 1995, and ending June 30, 1996.

```
title2 'Define a range inside the data range ';
title3 'Valid trading dates (19890131--19981231)';
title4 'My range is ( 19950101-19960630 )';

libname testit1 sasecrsp '/mydata/crspeng/m_sampdata/'
   setid=20
   permno=81871
   permno=82200
   permno=82224
   permno=83435
   permno=83696
   permno=83776
   permno=84788
   range='19950101-19960630';

data a;
   set testit1.ask;
run;

proc print data=a;
run;
```

Output 5.1.1. Printout of the ASK Monthly Time Series Data with RANGE

```
                                The SAS System                                 1
                       Define a range inside the data range
                      Valid trading dates (19890131--19981231)
                         My range is ( 19950101-19960630 )

                  Obs      PERMNO    CALDT              ASK

                    1       81871    19950731       18.25000
                    2       81871    19950831       19.25000
                    3       81871    19950929       26.00000
                    4       81871    19951031       26.00000
                    5       81871    19951130       25.50000
                    6       81871    19951229       24.25000
                    7       81871    19960131       22.00000
                    8       81871    19960229       32.50000
                    9       81871    19960329       30.25000
                   10       81871    19960430       33.75000
                   11       81871    19960531       27.50000
                   12       81871    19960628       30.50000
                   13       82200    19950831       49.50000
                   14       82200    19950929       62.75000
                   15       82200    19951031       88.00000
                   16       82200    19951130      138.50000
                   17       82200    19951229      139.25000
                   18       82200    19960131      164.25000
                   19       82200    19960229       51.00000
                   20       82200    19960329       41.62500
                   21       82200    19960430       61.25000
                   22       82200    19960531         .
                   23       82200    19960628       62.50000
                   24       82224    19950929       46.50000
                   25       82224    19951031       48.50000
                   26       82224    19951130       47.75000
                   27       82224    19951229       49.75000
                   28       82224    19960131       49.00000
                   29       82224    19960229       47.00000
                   30       82224    19960329       53.00000
                   31       82224    19960430       55.50000
                   32       82224    19960531         .
                   33       82224    19960628       51.00000
                   34       83435    19960430       30.25000
                   35       83435    19960531         .
                   36       83435    19960628       21.00000
                   37       83696    19960628       19.12500
```

Example 5.2. Extracting All PERMNOs Using a RANGE

If you want all PERMNOs extracted for the ASK time series from the monthly data set, then you would not specify the PERMNO= option.

```
title2 'Define a range inside the data range ';
title3 'Valid trading dates (19890131--19981231)';
title4 'My range is ( 19950101-19950228 )';

libname testit2 sasecrsp '/mydata/crspeng/m_sampdata/'
   setid=20
   range='19950101-19950228';

data b;
   set testit2.ask;
run;

proc print data=b;
run;
```

Output 5.2.1. Printout of All PERMNOs of ASK Monthly with RANGE

```
                              The SAS System                             1
                 Define a range inside the data range
                Valid trading dates (19890131--19981231)
                   My range is ( 19950101-19950228 )

                  Obs      PERMNO    CALDT               ASK

                    1       10078    19950131       32.75000
                    2       10078    19950228       32.12500
                    3       10104    19950131       42.87500
                    4       10104    19950228       31.50000
                    5       10107    19950131       59.37500
                    6       10107    19950228       63.00000
                    7       10155    19950131        2.43750
                    8       10155    19950228        3.00000
                    9       10837    19950131        3.12500
                   10       10837    19950228        3.50000
                   11       11042    19950131       21.87500
                   12       11042    19950228       23.50000
                   13       11081    19950131       42.62500
                   14       11081    19950228       41.62500
                   15       14593    19950131       40.50000
                   16       14593    19950228       39.62500
                   17       40484    19950131       35.75000
                   18       40484    19950228       38.50000
                   19       44194    19950131       14.25000
                   20       44194    19950228       15.25000
                   21       49606    19950131      139.00000
                   22       49606    19950228      129.50000
                   23       50156    19950131       44.75000
                   24       50156    19950228       41.75000
                   25       50404    19950131       18.37500
                   26       50404    19950228       20.12500
                   27       59248    19950131       16.75000
                   28       59248    19950228       16.25000
                   29       59328    19950131       69.37500
                   30       59328    19950228       79.75000
                   31       75030    19950131       21.25000
                   32       75030    19950228       22.87500
                   33       76535    19950131        2.75000
                   34       76535    19950228        2.75000
                   35       76829    19950131       21.50000
                   36       76829    19950228       25.00000
                   37       77352    19950131       15.25000
                   38       77352    19950228       16.00000
                   39       77418    19950131       54.50000
                   40       77418    19950228       71.25000
                   41       77882    19950131        6.50000
                   42       77882    19950228        7.12500
                   43       78083    19950131       32.75000
                   44       78083    19950228       34.75000
                   45       78117    19950131       39.25000
                   46       78117    19950228       39.00000
                   47       78987    19950131       22.50000
                   48       78987    19950228       25.25000
                   49       81181    19950131        3.75000
                   50       81181    19950228        3.62500
                   51       91708    19950131         .
                   52       91708    19950228         .
```

Example 5.3. Extracting One PERMNO Using No RANGE, Wide Open

If you want the entire range of available data for one particular PERMNO extracted from the monthly data set, then you would not specify the RANGE= option.

```
title2 'Select only PERMNO = 81871';
title3 'Valid trading dates (19890131--19981231)';
title4 'No range option, leave wide open';

libname testit3 sasecrsp '/mydata/crspeng/m_sampdata/'
   setid=20
   permno=81871;

data c;
   set testit3.ask;
run;

proc print data=c;
run;
```

Output 5.3.1. Printout of PERMNO = 81871 of ASK Monthly without RANGE

```
                              The SAS System                              1
                        Select only PERMNO = 81871
                  Valid trading dates (19890131--19981231)
                      No range option, leave wide open

                  Obs      PERMNO    CALDT              ASK

                    1       81871    19950731       18.25000
                    2       81871    19950831       19.25000
                    3       81871    19950929       26.00000
                    4       81871    19951031       26.00000
                    5       81871    19951130       25.50000
                    6       81871    19951229       24.25000
                    7       81871    19960131       22.00000
                    8       81871    19960229       32.50000
                    9       81871    19960329       30.25000
                   10       81871    19960430       33.75000
                   11       81871    19960531       27.50000
                   12       81871    19960628       30.50000
                   13       81871    19960731       26.12500
                   14       81871    19960830       19.12500
                   15       81871    19960930       19.50000
                   16       81871    19961031       14.00000
                   17       81871    19961129       18.75000
                   18       81871    19961231       24.25000
                   19       81871    19970131       29.75000
                   20       81871    19970228       24.37500
                   21       81871    19970331       15.00000
                   22       81871    19970430       18.25000
                   23       81871    19970530       25.12500
                   24       81871    19970630       31.12500
                   25       81871    19970731       35.00000
                   26       81871    19970829       33.00000
                   27       81871    19970930       26.81250
                   28       81871    19971031       18.37500
                   29       81871    19971128       16.50000
                   30       81871    19971231       16.25000
                   31       81871    19980130       22.75000
                   32       81871    19980227       21.00000
                   33       81871    19980331       22.50000
                   34       81871    19980430       16.12500
                   35       81871    19980529       11.12500
                   36       81871    19980630       13.43750
                   37       81871    19980731       22.87500
                   38       81871    19980831       17.75000
                   39       81871    19980930       24.25000
                   40       81871    19981030       26.00000
```

Example 5.4. Extracting Selected PERMNOs in Range Using INSET= Option

You can select the PERMNOs you want to extract and specify the range of data to extract by defining an INSET such as testin2.

```
title2 'INSET=testin2 uses date ranges along with PERMNOs:';
title3 '10107, 12490, 14322, 25788';
title4 'Begin dates and end dates for each permno are used in the INSET';

data testin2;
   permno = 10107; date1 = 19980731; date2 = 19981231; output;
   permno = 12490; date1 = 19970101; date2 = 19971231; output;
   permno = 14322; date1 = 19950731; date2 = 19960131; output;
   permno = 25778; date1 = 19950101; date2 = 19950331; output;
run;

libname mstk2 sasecrsp '/mydata/crspeng/m_sampdata/' setid=20
   inset='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
   set mstk2.prc;
run;

proc print data=b;
run;
```

Output 5.4.1. Printout of PRC Monthly Time Series Using INSET= Option

```
                                The SAS System                              1
          Second INSET=testin2 uses date ranges along with PERMNOs:
                          10107, 12490, 14322, 25788
       Begin dates and end dates for each permno are used in the INSET

                    Obs      PERMNO    CALDT               PRC

                      1       10107    19980731       109.93750
                      2       10107    19980831        95.93750
                      3       10107    19980930       110.06250
                      4       10107    19981030       105.87500
                      5       10107    19981130       122.00000
                      6       10107    19981231       138.68750
                      7       12490    19970131       156.87500
                      8       12490    19970228       143.75000
                      9       12490    19970331       137.25000
                     10       12490    19970430       160.50000
                     11       12490    19970530        86.50000
                     12       12490    19970630        90.25000
                     13       12490    19970731       105.75000
                     14       12490    19970829       101.37500
                     15       12490    19970930       106.00000
                     16       12490    19971031        98.50000
                     17       12490    19971128       109.50000
                     18       12490    19971231       104.62500
                     19       14322    19950731        32.62500
                     20       14322    19950831        32.37500
                     21       14322    19950929        36.87500
                     22       14322    19951031        34.00000
                     23       14322    19951130        39.37500
                     24       14322    19951229        39.00000
                     25       14322    19960131        41.50000
                     26       25778    19950131        49.87500
                     27       25778    19950228        57.25000
                     28       25778    19950331        59.37500
```

If you prefer, you can use PERMCO, or CUSIP, or HCUSIP, or TICKER in your inset= option instead of PERMNO to subset if you have them defined in your inset data set.

Example 5.5. Converting Dates Using the CRSP Date Functions

This example shows how to use the CRSP date functions and formats. The CRSPDTD formats are used for all the crspdt variables, while the YYMMDD format is used for the sasdt variables.

```
title2 'OUT= Data Set';
title3 'CRSP Functions for sasecrsp';
title4 'Always assign the libname sasecrsp first';

libname mstk sasecrsp '/mydata/crspeng/m_sampdata/' setid=20;

data a (keep = crspdt crspdt2 crspdt3
               sasdt sasdt2 sasdt3
               intdt intdt2 intdt3);
   format crspdt crspdt2 crspdt3 crspdtd8.;
   format sasdt sasdt2 sasdt3 yymmdd6.;
   format intdt intdt2 intdt3 8.;
   format exact 2.;
   crspdt = 1;
   sasdt = '2jul1962'd;
   intdt = 19620702;
   exact = 0;

/* Call the CRSP date to Integer function*/
   intdt2 = crspdcid(crspdt);

/* Call the SAS date to Integer function*/
   intdt3 = crspds2i(sasdt);

/* Call the Integer to Crsp date function*/
   crspdt2 = crspdicd(intdt,exact);

/* Call the Sas date to Crsp date conversion function*/
   crspdt3 = crspdscd(sasdt,exact);

/* Call the CRSP date to SAS date conversion function*/
   sasdt2 = crspdcsd(crspdt);

/* Call the Integer to Sas date conversion function*/
   sasdt3 = crspdi2s(intdt);
   run;

proc print;
   run;

title2 'Proc CONTENTS showing formats for sasecrsp';
proc contents data=a;
   run;
```

Output 5.5.1. Printout of Date Conversions Using the CRSP Date Functions

```
                                The SAS System                                 1
                                OUT= Data Set
                         CRSP Functions for sasecrsp
                   Always assign the libname sasecrsp first

Obs crspdt   crspdt2  crspdt3   sasdt sasdt2 sasdt3    intdt   intdt2   intdt3

 1  19620702 19620702 19620702 620702 620702 620702 19620702 19620702 19620702

                  Proc CONTENTS showing formats for sasecrsp
                         CRSP Functions for sasecrsp
                   Always assign the libname sasecrsp first

                            The CONTENTS Procedure

Data Set Name: WORK.A                                Observations:         1
Member Type:   DATA                                  Variables:            9
Engine:        V8                                    Indexes:              0
Created:       13:15 Monday, November 27, 2000       Observation Length:   72
Last Modified: 13:15 Monday, November 27, 2000       Deleted Observations: 0
Protection:                                          Compressed:           NO
Data Set Type:                                       Sorted:               NO
Label:


                 -----Engine/Host Dependent Information-----

       Data Set Page Size:         8192
       Number of Data Set Pages:   1
       First Data Page:            1
       Max Obs per Page:           113
       Obs in First Data Page:     1
       Number of Data Set Repairs: 0
       File Name:                  /tmp/SAS_work67560000352F/a.sas7bdat
       Release Created:            8.0202M0
       Host Created:               HP-UX
       Inode Number:               414
       Access Permission:          rw-r--r--
       Owner Name:                 saskff
       File Size (bytes):          16384


             -----Alphabetic List of Variables and Attributes-----

               #    Variable    Type    Len    Pos    Format
               ------------------------------------------------
               1    crspdt      Num       8      0    CRSPDTD8.
               2    crspdt2     Num       8      8    CRSPDTD8.
               3    crspdt3     Num       8     16    CRSPDTD8.
               7    intdt       Num       8     48    8.
               8    intdt2      Num       8     56    8.
               9    intdt3      Num       8     64    8.
               4    sasdt       Num       8     24    YYMMDD6.
               5    sasdt2      Num       8     32    YYMMDD6.
               6    sasdt3      Num       8     40    YYMMDD6.
```

References

Center for Research in Security Prices (2000), *CRSP Data Description Guide*, Chicago: The University of Chicago Graduate School of Business, [http://www.crsp.uchicago.edu/file_guides/stock_ind_data_descriptions.pdf].

Center for Research in Security Prices (2000), *CRSP Programmer's Guide*, Chicago: The University of Chicago Graduate School of Business, [http://www.crsp.uchicago.edu/file_guides/stock_ind_programming.pdf].

Center for Research in Security Prices (2000), *CRSPAccess Database Format Release Notes*, Chicago: The University of Chicago Graduate School of Business, [http://www.crsp.uchicago.edu/file_guides/ca_release_notes.pdf].

Center for Research in Security Prices (2000), *CRSP Utilities Guide*, Chicago: The University of Chicago Graduate School of Business, [http://www.crsp.uchicago.edu/file_guides/stock_ind_utilities.pdf].

Center for Research in Security Prices (2000), *CRSP SFA Guide*, Chicago: The University of Chicago Graduate School of Business, [http://www.crsp.uchicago.edu/file_guides/stock_ind_sfa.pdf].

Acknowledgments

Many people have been instrumental in the development of the ETS Interface engine. The individuals listed here have been especially helpful.

Bob Spatz, Center for Research in Security Prices, University of Chicago Graduate School of Business.

Ken Kraus, Center for Research in Security Prices, University of Chicago Graduate School of Business.

Rick Langston, SAS Institute, Cary, NC.

Peng Zang, University of Illinois, Urbana, IL.

Stephen Caudle, Rose-Hulman Institute of Technology, Terre Haute, IN.

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 6
The SASEFAME Interface Engine

Chapter Contents

Chapter 6
The SASEFAME Interface Engine

Overview

The SASEFAME interface engine enables SAS users to access and process time series data residing in a FAME database, and provides a seamless interface between FAME and SAS data processing.

The SASEFAME engine uses the LIBNAME statement to enable you to specify which time series you would like to read from the FAME database, and how you would like to convert the selected time series to the same time scale. The SAS DATA step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time.

SASEFAME for Release 8.2 supports Windows, Solaris2, AIX, and HP-UX hosts.

SASEFAME for Version 9 supports Windows, Solaris 8, AIX, LINUX and DEC/OSF Digital UNIX.

Getting Started

Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASEFAME engine provides a time ID variable named DATE.

Reading and Converting FAME Database Time Series

The SASEFAME engine supports reading and converting time series stored in FAME databases. The SASEFAME engine uses the FAME WORK database to temporarily store the converted time series. All series specified by the FAME wildcard are written to the FAME WORK database. For conversion of very large databases, you may want to define the FAME_TEMP environment variable to point to a location where there is ample space for FAME WORK. The FAME CHLI does not support writing more than 2 gigabytes to the FAME WORK area, and when this happens SASEFAME will terminate with a system error.

Using the SAS DATA Step

If desired, you can store the converted series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once your data is stored in a SAS data set you can use it as you would any other SAS data set.

Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information concerning the contents of your data set by using the CONTENTS procedure, as in Example 6.1. You can create a view of the FAME database by using the SQL procedure to create your view using the SASEFAME engine in your *libref*, along with the using clause. See Example 6.5.

Using Remote FAME Data Access

There are two ways to access your remote FAME databases. The first way is to use your SAS/CONNECT capability to submit a remote SAS session to your FAME server, and then use PROC DOWNLOAD to bring your results back to your SAS client. See Example 6.6. Refer to *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

The second way is an experimental feature of the SASEFAME interface that uses the FAME CHLI to communicate with your remote FAME server by giving the frdb_m port number and node name of your FAME master server in your *libref*. See Example 6.7. Refer to “Starting the Master Server” in the *Guide to FAME Database Servers*.

Syntax

The SASEFAME engine uses standard engine syntax. Options used by SASEFAME are summarized in the table below.

Description	**Statement**	**Option**
specifies the FAME frequency and the FAME technique.	LIBNAME *libref* SASEFAME	CONVERT=
specifies a FAME wildcard to match data object series names within the FAME database, which limits the selection of time series that are included in the SAS data set.	LIBNAME *libref* SASEFAME	WILDCARD=

The LIBNAME *libref* SASEFAME Statement

LIBNAME *libref* **SASEFAME ’physical name’** *options;*

Since *physical name* specifies the location of the folder where your FAME database resides, it should end in a backslash if you are on a Windows platform, or a forward slash if you are on a UNIX platform.

If you are accessing a remote FAME database using FAME CHLI, you can use the following syntax for *physical name*:
’**#port number\@hostname physical path name**’

The following options can be used in the LIBNAME libref SASEFAME statement:

CONVERT=*(FREQ=fame_frequency TECH= fame_technique)*

specifies the FAME frequency and the FAME technique just as you would in the FAME CONVERT function. There are four possible values for *fame_technique*: CONSTANT (default), CUBIC, DISCRETE, or LINEAR. All FAME frequencies except PPY and YPP are supported by the SASEFAME engine. For a more complete discussion of FAME frequencies and SAS time intervals see the section “Mapping FAME Frequencies to SAS Time Intervals” on page 190. For all possible *fame_frequency* values, refer to “Understanding Frequencies” in the *User’s Guide to FAME*. As an example,

```
LIBNAME libref sasefame ’physical-name’
   CONVERT=(TECH=CONSTANT FREQ=TWICEMONTHLY);
```

WILDCARD=*"fame_wildcard"*

By default, the SASEFAME engine reads all time series in the FAME database that you name in your SASEFAME libref. You can limit the time series read from the FAME database by specifying the WILDCARD= option on your LIBNAME statement. The *fame_wildcard* is a quoted string containing the FAME wildcard you wish to use. The wildcard is used against the data object names (time series only) in the FAME database that resides in the library you are in the process of assigning. For more information about wildcarding, see “Specifying Wildcards” in the *User’s Guide to FAME*.

For example, to read all time series in the TEST library being accessed by the SASEFAME engine, you would specify

```
LIBNAME test sasefame ’physical name of test database’
   WILDCARD="?";
```

To read series with names such as A_DATA, B_DATA, C_DATA, you could specify

```
LIBNAME test sasefame ’physical name of test database’
   WILDCARD="^_DATA";
```

When you use the WILD= option, you are limiting the number of time series that are read and converted to the desired frequency. This option can help you save resources when processing large databases or when processing a large number of observations, such as daily, hourly, or minutely frequencies. Since the SASEFAME engine uses the FAME WORK database to store the converted time series, using wildcards is recommended to prevent your WORK space from getting too large.

Details

The SAS Output Data Set

You can use the SAS DATA step to write the FAME converted series to a SAS data set. This allows the user the ability to easily analyze the data using SAS. You can specify the name of the output data set on the DATA statement. This causes the engine supervisor to create a SAS data set using the specified name in either the SAS WORK library, or if specified, the USER library. For more about naming your SAS data set see the section “Characteristics of SAS Data Libraries” in *SAS Language Reference: Dictionary*.

The contents of the SAS data set include the DATE of each observation, the name of each series read from the FAME database as specified by the WILDCARD option, and the label or FAME description of each series. Missing values are represented as ’.’ in the SAS data set. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL along with the SASEFAME engine to create a view of your SAS data set.

The DATE variable in the SAS data set contains the date of the observation. For FAME weekly intervals that end on a Friday, FAME reports the date on the Friday that ends the week whereas SAS reports the date on the Saturday that begins the week. A more detailed discussion of how to map FAME frequencies to SAS Time Intervals follows.

Mapping FAME Frequencies to SAS Time Intervals

The following table summarizes the mapping of FAME frequencies to SAS time intervals. It is important to note that FAME frequencies often have a sample unit in parentheses following the keyword frequency. This sample unit is an end-of-interval unit. SAS dates are represented using begin-of-interval notation. For more on SAS time intervals, see “Date Intervals, Formats, and Functions” in *SAS/ETS User’s Guide*. For more on FAME frequencies, see the section “Understanding Frequencies” in the *User’s Guide to FAME*.

FAME FREQUENCY	SAS TIME INTERVAL
WEEKLY (SUNDAY)	WEEK.2
WEEKLY (MONDAY)	WEEK.3
WEEKLY (TUESDAY)	WEEK.4
WEEKLY (WEDNESDAY)	WEEK.5

FAME FREQUENCY	SAS TIME INTERVAL
WEEKLY (THURSDAY)	WEEK.6
WEEKLY (FRIDAY)	WEEK.7
WEEKLY (SATURDAY)	WEEK.1
BIWEEKLY (ASUNDAY)	WEEK2.2
BIWEEKLY (AMONDAY)	WEEK2.3
BIWEEKLY (ATUESDAY)	WEEK2.4
BIWEEKLY (AWEDNESDAY)	WEEK2.5
BIWEEKLY (ATHURSDAY)	WEEK2.6
BIWEEKLY (AFRIDAY)	WEEK2.7
BIWEEKLY (ASATURDAY)	WEEK2.1
BIWEEKLY (BSUNDAY)	WEEK2.9
BIWEEKLY (BMONDAY)	WEEK2.10
BIWEEKLY (BTUESDAY)	WEEK2.11
BIWEEKLY (BWEDNESDAY)	WEEK2.12
BIWEEKLY (BTHURSDAY)	WEEK2.13
BIWEEKLY (BFRIDAY)	WEEK2.14
BIWEEKLY (BSATURDAY)	WEEK2.8
BIMONTHLY (NOVEMBER)	MONTH2.2
BIMONTHLY	MONTH2.1
QUARTERLY (OCTOBER)	QTR.2
QUARTERLY (NOVEMBER)	QTR.3
QUARTERLY	QTR.1
ANNUAL (JANUARY)	YEAR.2
ANNUAL (FEBRUARY)	YEAR.3
ANNUAL (MARCH)	YEAR.4
ANNUAL (APRIL)	YEAR.5
ANNUAL (MAY)	YEAR.6
ANNUAL (JUNE)	YEAR.7
ANNUAL (JULY)	YEAR.8
ANNUAL (AUGUST)	YEAR.9
ANNUAL (SEPTEMBER)	YEAR.10
ANNUAL (OCTOBER)	YEAR.11
ANNUAL (NOVEMBER)	YEAR.12
ANNUAL	YEAR.1
SEMIANNUAL (JULY)	SEMIYEAR.2
SEMIANNUAL (AUGUST)	SEMIYEAR.3
SEMIANNUAL (SEPTEMBER)	SEMIYEAR.4
SEMIANNUAL (OCTOBER)	SEMIYEAR.5
SEMIANNUAL (NOVEMBER)	SEMIYEAR.6
SEMIANNUAL	SEMIYEAR.1

FAME FREQUENCY	SAS TIME INTERVAL
YPP	not supported
PPY	not supported
SECONDLY	SECOND
MINUTELY	MINUTE
HOURLY	HOUR
DAILY	DAY
BUSINESS	WEEKDAY
TENDAY	TENDAY
TWICEMONTHLY	SEMIMONTH
MONTHLY	MONTH

Examples

Example 6.1. Converting an Entire FAME Database

To enable conversion of all Time Series no wildcard is specified, so the default “?” wildcard is used. Always consider both the number of time series and the number of observations generated by the conversion process. The converted series are stored in the FAME WORK database during the SAS DATA step. You may further limit your resulting SAS data set by using KEEP, DROP, or WHERE statements inside your data step.

The following statements convert a FAME database and print out its contents:

```
libname famedir sasefame '.'
                convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* do nothing special */
   run;

proc print data=mydir.a; run;
```

In the above example, the FAME database is called oecd1.db and it resides in the famedir directory. The DATA statement names the SAS output data set ’a’ which will reside in mydir. All time series in the FAME oecd1.db database will be converted to an annual frequency and stored in the mydir.a SAS data set. The PROC PRINT statement creates a listing of all of the observations in the mydir.a SAS data set.

Example 6.2. Reading Time Series from the FAME Database

Use the FAME wildcard option to limit the number of series converted. For example, suppose you want to read only series starting with “WSPCA”. You could use the following code:

```
libname lib1 sasefame '/mine/data/econ_fame/sampdir'
             wildcard="wspca?"
             convert=(technique=constant freq=twicemonthly );

libname lib2 '/mine/data/econ_sas/sampdir';

data lib2.twild(label='Annual Series from the FAMEECON.db');
   set lib1.subecon;
   /* keep only  */
   keep date wspca;
   run;

proc contents data=lib2.twild; run;

proc print data=lib2.twild; run;
```

The wildcard=“wspca?” option limits reading only those series whose names begin with WSPCA. The SAS KEEP statement further restricts the SAS data set to include only the series named WSPCA and the DATE variable. The time interval used for the conversion is TWICEMONTHLY.

Example 6.3. Writing Time Series to the SAS Data Set

You can use the KEEP or DROP statement to include or exclude certain series names from the SAS data set.

```
libname famedir sasefame '.'
        convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   drop ita_dird--jpn_herd tur_dird--usa_herd;
   run;

proc print data=mydir.a; run;
```

You can rename your SAS variables by using the RENAME statement.

```
option validvarname=any;

libname famedir sasefame '.'
```

```
        convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* keep and rename  */
   keep date ita_dird--jpn_herd tur_dird--usa_herd;
   rename ita_dird='Italy.dirdes'n
          jpn_dird='Japan.dirdes'n
          tur_dird='Turkey.dirdes'n
          usa_dird='UnitedStates.dirdes'n ;
   run;

proc print data=mydir.a; run;
```

Example 6.4. Limiting the Time Range of Data

You may also limit the time range of the data in the SAS data set by using the WHERE statement in the data step to process the time ID variable DATE only when it falls in the range you are interested in.

```
libname famedir SASEFAME '.'
        convert=(freq=annual technique=constant);

libname mydir '/mine/data/europe/sas/oecdir';

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* where only  */
   where date between '01jan88'd and '31dec90'd;
   run;

proc print data=mydir.a; run;
```

All data for 1988, 1989, and 1990 are included in the SAS data set. See the *SAS Language: Reference, Version 7* for more information on KEEP, DROP, RENAME and WHERE statements.

Example 6.5. Creating a View Using the SQL Procedure and SASEFAME

This example creates a view using the SQL procedure's from and using clauses. See *SQL Procedure Guide, Version 7* for details on SQL views.

```
title1 'famesql5: PROC SQL Dual Embedded Libraries w/ FAME option';
options validvarname=any;

/* Dual Embedded Library Allocations (With FAME Option) */
/********************************************************/

/* OECD1 Database */
/******************/

title2 'OECD1: Dual Embedded Library Allocations
                  with FAME Option';
proc sql;
   create view fameview as
   select date, 'fin.herd'n
   from lib1.oecd1
   using libname lib1 sasefame '/economic/databases/testdat'
                      convert=(tech=constant freq=annual),
         libname temp '/usr/local/scratch/mine'
   quit;

title2 'OECD1: Print of View from Embedded Library
                  with FAME Option';
proc print data=fameview;
run;
```

Output 6.5.1. Printout of the FAME View of OECD Data

```
              PROC SQL Dual Embedded Librarys w/ FAME option
          OECD1: Print of View from Embedded Library with Option

                         Obs     DATE     FIN.HERD

                          1      1985      1097.0
                          2      1986      1234.0
                          3      1987      1401.3
                          4      1988      1602.0
                          5      1989      1725.5
                          6      1990      1839.0
                          7      1991         .
```

```
/* SUBECON Database */
/*********************/

title2 'SUBECON: Dual Embedded Library Allocations
                 with FAME Option';
proc sql;
   create view fameview as
   select date, gaa
   from lib1.subecon
   using libname lib1 sasefame '/economic/databases/testdat'
                      convert=(tech=constant freq=annual),
         libname temp '/usr/local/scratch/mine'
   quit;

title2 'SUBECON: Print of View from Embedded Library
                      with FAME Option';
proc print data=fameview;
run;
```

Output 6.5.2. Printout of the FAME View of DRI Basic Economic Data

```
                PROC SQL Dual Embedded Librarys w/ FAME option
           SUBECON: Print of View from Embedded Library with Option

                          Obs    DATE        GAA

                            1    1946          .
                            2    1947          .
                            3    1948     23174.00
                            4    1949     19003.00
                            5    1950     24960.00
                            6    1951     21906.00
                            7    1952     20246.00
                            8    1953     20912.00
                            9    1954     21056.00
                           10    1955     27168.00
                           11    1956     27638.00
                           12    1957     26723.00
                           13    1958     22929.00
                           14    1959     29729.00
                           15    1960     28444.00
                           16    1961     28226.00
                           17    1962     32396.00
                           18    1963     34932.00
                           19    1964     40024.00
                           20    1965     47941.00
                           21    1966     51429.00
                           22    1967     49164.00
                           23    1968     51208.00
                           24    1969     49371.00
                           25    1970     44034.00
                           26    1971     52352.00
                           27    1972     62644.00
                           28    1973     81645.00
                           29    1974     91028.00
                           30    1975     89494.00
                           31    1976    109492.00
                           32    1977    130260.00
                           33    1978    154357.00
                           34    1979    173428.00
                           35    1980    156096.00
                           36    1981    147765.00
                           37    1982    113216.00
                           38    1983    133495.00
                           39    1984    146448.00
                           40    1985    128521.99
                           41    1986    111337.99
                           42    1987    160785.00
                           43    1988    210532.00
                           44    1989    201637.00
                           45    1990    218702.00
                           46    1991    210666.00
                           47    1992          .
                           48    1993          .
```

```
title2 'DB77: Dual Embedded Library Allocations
                 with FAME Option';
proc sql;
   create view fameview as
   select date, ann, 'qandom.x'n
   from lib1.db77
   using libname lib1 sasefame '/economic/databases/testdat'
                      convert=(tech=constant freq=annual),
         libname temp '/usr/local/scratch/mine'
   quit;

  title2 'DB77: Print of View from Embedded Library
                   with FAME Option';
  proc print data=fameview;
  run;
```

Output 6.5.3. Printout of the FAME View of DB77 Data

```
              PROC SQL Dual Embedded Librarys w/ FAME option
           DB77: Print of View from Embedded Library with Option

                     Obs    DATE    ANN    QANDOM.X

                       1    1959      .     0.56147
                       2    1960      .     0.51031
                       3    1961      .      .
                       4    1962      .      .
                       5    1963      .      .
                       6    1964      .      .
                       7    1965      .      .
                       8    1966      .      .
                       9    1967      .      .
                      10    1968      .      .
                      11    1969      .      .
                      12    1970      .      .
                      13    1971      .      .
                      14    1972      .      .
                      15    1973      .      .
                      16    1974      .      .
                      17    1975      .      .
                      18    1976      .      .
                      19    1977      .      .
                      20    1978      .      .
                      21    1979      .      .
                      22    1980    100      .
                      23    1981    101      .
                      24    1982    102      .
                      25    1983    103      .
                      26    1984    104      .
                      27    1985    105      .
                      28    1986    106      .
                      29    1987    107      .
                      30    1988    109      .
                      31    1989    111      .
```

```
/* DRIECON Database */
/********************/

title2 'DRIECON: Dual Embedded Library Allocations
                 with FAME Option';
proc sql;
   create view fameview as
   select date, husts
   from lib1.driecon
   using libname lib1 sasefame '/economic/databases/testdat'
                      convert=(tech=constant freq=annual),
         libname temp '/usr/local/scratch/mine'
   quit;

title2 'DRIECON: Print of View from Embedded Library
                 with FAME Option';
proc print data=fameview;
run;
```

Note that the SAS option VALIDVARNAME=ANY was used at the top of this example due to special characters being present in the time series names. The output from this example shows how each FAME view is the output of the SASEFAME engine's processing. Note that different engine options could have been used in the USING LIBNAME clause if desired.

Output 6.5.4. Printout of the FAME View of DRI Basic Economic Data

```
          PROC SQL Dual Embedded Librarys w/ FAME option
     DRIECON: Print of View from Embedded Library with Option

                    Obs    DATE     HUSTS

                      1    1947    1.26548
                      2    1948    1.33470
                      3    1949    1.43617
                      4    1950    1.90041
                      5    1951    1.43759
                      6    1952    1.44883
                      7    1953    1.40279
                      8    1954    1.53525
                      9    1955    1.61970
                     10    1956    1.32400
                     11    1957    1.17300
                     12    1958    1.31717
                     13    1959    1.53450
                     14    1960    1.25505
                     15    1961    1.31188
                     16    1962    1.45996
                     17    1963    1.58858
                     18    1964    1.53950
                     19    1965    1.46966
                     20    1966    1.16507
                     21    1967    1.28573
                     22    1968    1.50314
                     23    1969    1.48531
                     24    1970    1.43565
                     25    1971    2.03775
                     26    1972    2.36069
                     27    1973    2.04307
                     28    1974    1.32855
                     29    1975    1.16164
                     30    1976    1.53468
                     31    1977    1.96218
                     32    1978    2.00184
                     33    1979    1.71847
                     34    1980    1.29990
                     35    1981    1.09574
                     36    1982    1.05862
                     37    1983    1.70580
                     38    1984    1.76351
                     39    1985    1.74258
                     40    1986    1.81205
                     41    1987    1.62914
                     42    1988    1.48748
                     43    1989    1.38218
                     44    1990    1.20161
                     45    1991    1.00878
                     46    1992    1.20159
                     47    1993    1.29201
                     48    1994    1.44684
                     49    1995    1.35845
                     50    1996    1.48336
```

Example 6.6. Remote FAME Access, using SAS CONNECT

Suppose you are running SAS in a client/server environment and have SAS/CONNECT capability allowing you access to your FAME server. You could access your FAME remote data by signing on to your Fame server from your client session and doing a remote submit to access your Fame data. You could then use PROC DOWNLOAD to bring your remote data into the local client SAS session.

```
options validvarname=any;

%let remnode=mysunbox.unx.sas.com;
signon remnode.shr2;  /* name the sastcpd service for the connection */
rsubmit;

%let FAME=%sysget(FAME);
%put(&FAME);

options validvarname=any;
libname lib1 sasefame "/usr/local/famelib/util"
             convert=(frequency=annual technique=constant)
             wildcard="?";

data oecd1;
   set lib1.oecd1;
run;

title2 'OECD1: PRINT Procedure';
proc print data=oecd1(obs=10);
run;

title2 'OECD1: CONTENTS Procedure';
proc contents data=oecd1;
run;

proc sort data=oecd1;
   by date;
run;

title2 'OECD1: MEANS Procedure';
proc means data=oecd1 sum;
   by date;
   var 'fin.herd'n;
run;

proc download inlib=work  /* remote SASWORK */
   outlib=work;  /* local SASWORK */
run;


proc datasets lib=work;
   contents data=_ALL_;
run;
```

```
endrsubmit;
signoff;

proc datasets lib=work;  /* NOW the local work has the fame data */
   contents data=_ALL_;
run;
```

Output 6.6.1. Printout of the SAS/CONNECT Remote Access to FAME Data

```
          ***fameproc: Using FAME with a multitude of SAS Procs***
                           OECD1: PRINT Procedure

           AUS.             AUT.             BEL.             CAN.  CAN.
Obs DATE  DIRDES AUS.HERD  DIRDES AUT.HERD  DIRDES BEL.HERD  DIRDES HERD

 1  1985    .         .   360.900  5990.60 334.700 14936.00 1345.90 1642
 2  1986 680.400   881.70    .         .   341.300 15459.80 1435.30 1755
 3  1987 725.800   983.80    .         .   369.200 16614.30 1486.50 1849
 4  1988 750.000  1072.90    .         .   374.000 16572.70 1589.60 2006
 5  1989    .         .      .         .      .    18310.70 1737.00 2214
 6  1990    .         .      .         .      .    18874.20 1859.20 2347
 7  1991    .         .      .         .      .         .   1959.60 2488

      CHE.             DEU.             DNK.             ESP.
Obs  DIRDES CHE.HERD  DIRDES DEU.HERD  DIRDES DNK.HERD  DIRDES ESP.HERD

 1     .         .   2702.10  6695.70 191.300  1873.70 335.500  31987.0
 2  366.600    900       .    7120.00 211.700  2123.00 354.800  36778.0
 3     .         .   3365.60  8338.50 232.700  2372.00 409.900  43667.0
 4  632.100   1532   3538.60  8780.00 258.100  2662.00 508.200  55365.5
 5     .      1648   3777.20  9226.60 284.800  2951.00 623.600  69270.5
 6     .         .   2953.30  9700.00    .         .   723.600  78848.0
 7     .         .       .        .      .         .      .     89908.0

      FIN.             FRA.             GBR.             GRC.
Obs  DIRDES FIN.HERD  DIRDES FRA.HERD  DIRDES GBR.HERD  DIRDES GRC.HERD

 1  183.700  1097.00 2191.60 15931.00 2068.40  1174.00    .         .
 2  202.000  1234.00 2272.90 17035.00 2226.10  1281.00  44.600  3961.00
 3  224.300  1401.30 2428.70 18193.00 2381.10  1397.20    .         .
 4  247.700  1602.00 2573.50 19272.00 2627.00  1592.00  60.600  6674.50
 5  259.700  1725.50 2856.50 21347.80 2844.10  1774.20 119.800 14485.20
 6  271.000  1839.00 3005.20 22240.00    .         .      .         .
 7     .         .       .        .      .         .      .         .

      IRL.             ISL.             ITA.             JPN.
Obs  DIRDES IRL.HERD  DIRDES ISL.HERD  DIRDES ITA.HERD  DIRDES  JPN.HERD

 1  39.2000  28.3707  7.1000  268.300 1344.90  1751008  8065.70  1789780
 2  46.5000  35.0000   .         .    1460.60  2004453  8290.10  1832575
 3  48.1000  36.0380  7.8000  420.400 1674.40  2362102  9120.80  1957921
 4  49.6000  37.0730   .         .    1861.50  2699927  9657.20  2014073
 5  50.2000  39.0130 10.3000  786.762 1968.00  2923504 10405.90  2129372
 6  51.7000    .     11.0000  902.498 2075.00  3183071     .     2296992
 7    .        .     11.8000  990.865 2137.80  3374000     .           .

      NLD.             NOR.             NZL.             PRT.
Obs  DIRDES NLD.HERD  DIRDES NOR.HERD  DIRDES NZL.HERD  DIRDES PRT.HERD

 1  798.400   2032   209.000  1802.50 59.2000   80.100    .         .
 2  836.000   2093      .         .     .          .    76.700  5988.90
 3  886.500   2146   250.000  2165.80   .          .      .         .
 4  883.000   2105      .         .     .          .   111.500 10158.20
 5  945.000   2202   308.900  2771.40 78.7000  143.800    .         .
 6     .         .      .         .     .          .      .         .
 7     .         .   352.000  3100.00   .          .      .         .

      SWE.             TUR.             USA.              YUG.
Obs  DIRDES SWE.HERD  DIRDES TUR.HERD  DIRDES  USA.HERD  DIRDES YUG.HERD

 1   840.10     6844 144.800   22196  14786.00 14786.00 175.900     1.87
 2     .          .  136.400   26957  16566.90 16566.90 208.600     4.10
 3  1016.90     8821 121.900   32309  18326.10 18326.10 237.000    10.21
 4     .          .  174.400   74474  20246.20 20246.20 233.000    29.81
 5  1076.00    11104 212.300  143951  22159.50 22159.50 205.100   375.22
 6     .          .     .          .  23556.10 23556.10    .     2588.50
 7     .          .     .          .  24953.80 24953.80    .          .
```

Output 6.6.2. Proc CONTENTS of the Remote FAME Data on the SUN server node

```
                ***fameproc: Using FAME on Remote Server ***
                               OECD1: CONTENTS Procedure
                                 The CONTENTS Procedure

    Data Set Name: WORK.OECD1                              Observations:         7
    Member Type:   DATA                                    Variables:            49
    Engine:        V8                                      Indexes:              0
    Created:       13:19 Wednesday, May 10, 2000           Observation Length:   392
    Last Modified: 13:19 Wednesday, May 10, 2000           Deleted Observations: 0
    Protection:                                            Compressed:           NO
    Data Set Type:                                         Sorted:               NO
    Label:
                      -----Engine/Host Dependent Information-----

    Data Set Page Size:         32768
    Number of Data Set Pages:   1
    First Data Page:            1
    Max Obs per Page:           83
    Obs in First Data Page:     7
    Number of Data Set Repairs: 0
    File Name:                  /usr/tmp/SAS_work3EEB00002CC4_sunbox/oecd1.sas7bdat
    Release Created:            8.0101M0
    Host Created:               SunOS
    Inode Number:               211927040
    Access Permission:          rw-rw-rw-
    Owner Name:                 myuser
    File Size (bytes):          40960
```

Output 6.6.3. Proc CONTENTS of the Remote FAME Data on the SUN server node

```
                ***fameproc: Using FAME on Remote Server ***
                               OECD1: CONTENTS Procedure
                                 The CONTENTS Procedure

                -----Alphabetic List of Variables and Attributes-----

 #    Variable      Type    Len    Pos    Format  Informat    Label
-----------------------------------------------------------------------------
 2    AUS.DIRDES    Num       8      8
 3    AUS.HERD      Num       8     16
 4    AUT.DIRDES    Num       8     24
 5    AUT.HERD      Num       8     32
 6    BEL.DIRDES    Num       8     40
 7    BEL.HERD      Num       8     48
 8    CAN.DIRDES    Num       8     56
 9    CAN.HERD      Num       8     64
10    CHE.DIRDES    Num       8     72
11    CHE.HERD      Num       8     80
 1    DATE          Num       8      0    YEAR4.  4.        Date of Observation
12    DEU.DIRDES    Num       8     88
13    DEU.HERD      Num       8     96
14    DNK.DIRDES    Num       8    104
15    DNK.HERD      Num       8    112
16    ESP.DIRDES    Num       8    120
17    ESP.HERD      Num       8    128
18    FIN.DIRDES    Num       8    136
19    FIN.HERD      Num       8    144
20    FRA.DIRDES    Num       8    152
21    FRA.HERD      Num       8    160
22    GBR.DIRDES    Num       8    168
23    GBR.HERD      Num       8    176
24    GRC.DIRDES    Num       8    184
25    GRC.HERD      Num       8    192
26    IRL.DIRDES    Num       8    200
27    IRL.HERD      Num       8    208
28    ISL.DIRDES    Num       8    216
29    ISL.HERD      Num       8    224
30    ITA.DIRDES    Num       8    232
31    ITA.HERD      Num       8    240
32    JPN.DIRDES    Num       8    248
33    JPN.HERD      Num       8    256
34    NLD.DIRDES    Num       8    264
35    NLD.HERD      Num       8    272
36    NOR.DIRDES    Num       8    280
37    NOR.HERD      Num       8    288
38    NZL.DIRDES    Num       8    296
39    NZL.HERD      Num       8    304
40    PRT.DIRDES    Num       8    312
41    PRT.HERD      Num       8    320
42    SWE.DIRDES    Num       8    328
43    SWE.HERD      Num       8    336
44    TUR.DIRDES    Num       8    344
45    TUR.HERD      Num       8    352
46    USA.DIRDES    Num       8    360
47    USA.HERD      Num       8    368
48    YUG.DIRDES    Num       8    376
49    YUG.HERD      Num       8    384
```

Output 6.6.4. Proc MEANS and TABULATE of Remote FAME Data

```
       ***fameproc: Using FAME with a multitude of SAS Procs***
                        OECD1: MEANS Procedure

------------------ Date of Observation=1985 ------------------
                         The MEANS Procedure

                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1097.00
                        ------------
------------------ Date of Observation=1986 ------------------
                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1234.00
                        ------------
------------------ Date of Observation=1987 ------------------
                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1401.30
                        ------------
------------------ Date of Observation=1988 ------------------
                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1602.00
                        ------------
------------------ Date of Observation=1989 ------------------
                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1725.50
                        ------------
------------------ Date of Observation=1990 ------------------
                         The MEANS Procedure

                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                             1839.00
                        ------------
------------------ Date of Observation=1991 ------------------
                 Analysis Variable : FIN.HERD
                                 Sum
                        ------------
                                   .
                        ------------

                  OECD1: TABULATE Procedure

                       --------------
                       |  FIN.HERD  |
                       |------------|
                       |    Sum     |
                       |------------|
                       |     8898.80|
                       --------------
```

Output 6.6.5. Proc CONTENTS of the Remote FAME Data on the HP-UX client node

```
                              The DATASETS Procedure

    Data Set Name: WORK.OECD1                             Observations:         7
    Member Type:   DATA                                   Variables:            49
    Engine:        V8                                     Indexes:              0
    Created:       13:20 Wednesday, May 10, 2000          Observation Length:   392
    Last Modified: 13:20 Wednesday, May 10, 2000          Deleted Observations: 0
    Protection:                                           Compressed:           NO
    Data Set Type:                                        Sorted:               YES
    Label:

                      -----Engine/Host Dependent Information-----

          Data Set Page Size:         32768
          Number of Data Set Pages:   1
          First Data Page:            1
          Max Obs per Page:           83
          Obs in First Data Page:     7
          Number of Data Set Repairs: 0
          File Name:                  /tmp/SAS_workDF8500003D1D/oecd1.sas7bdat
          Release Created:            8.0202M0
          Host Created:               HP-UX
          Inode Number:               1452
          Access Permission:          rw-r--r--
          Owner Name:                 myuser
          File Size (bytes):          40960
```

Output 6.6.6. Proc CONTENTS of the Remote FAME Data on the HP-UX client node

```
           ***fameproc: Using FAME with a multitude of SAS Procs***           1
                               OECD1: CONTENTS Procedure
                                 The CONTENTS Procedure

                  -----Alphabetic List of Variables and Attributes-----

  #    Variable      Type    Len    Pos    Format  Informat    Label
 ---------------------------------------------------------------------------
  2    AUS.DIRDES    Num       8      8
  3    AUS.HERD      Num       8     16
  4    AUT.DIRDES    Num       8     24
  5    AUT.HERD      Num       8     32
  6    BEL.DIRDES    Num       8     40
  7    BEL.HERD      Num       8     48
  8    CAN.DIRDES    Num       8     56
  9    CAN.HERD      Num       8     64
 10    CHE.DIRDES    Num       8     72
 11    CHE.HERD      Num       8     80
  1    DATE          Num       8      0    YEAR4.  4.          Date of Observation
 12    DEU.DIRDES    Num       8     88
 13    DEU.HERD      Num       8     96
 14    DNK.DIRDES    Num       8    104
 15    DNK.HERD      Num       8    112
 16    ESP.DIRDES    Num       8    120
 17    ESP.HERD      Num       8    128
 18    FIN.DIRDES    Num       8    136
 19    FIN.HERD      Num       8    144
 20    FRA.DIRDES    Num       8    152
 21    FRA.HERD      Num       8    160
 22    GBR.DIRDES    Num       8    168
 23    GBR.HERD      Num       8    176
 24    GRC.DIRDES    Num       8    184
 25    GRC.HERD      Num       8    192
 26    IRL.DIRDES    Num       8    200
 27    IRL.HERD      Num       8    208
 28    ISL.DIRDES    Num       8    216
 29    ISL.HERD      Num       8    224
 30    ITA.DIRDES    Num       8    232
 31    ITA.HERD      Num       8    240
 32    JPN.DIRDES    Num       8    248
 33    JPN.HERD      Num       8    256
 34    NLD.DIRDES    Num       8    264
 35    NLD.HERD      Num       8    272
 36    NOR.DIRDES    Num       8    280
 37    NOR.HERD      Num       8    288
 38    NZL.DIRDES    Num       8    296
 39    NZL.HERD      Num       8    304
 40    PRT.DIRDES    Num       8    312
 41    PRT.HERD      Num       8    320
 42    SWE.DIRDES    Num       8    328
 43    SWE.HERD      Num       8    336
 44    TUR.DIRDES    Num       8    344
 45    TUR.HERD      Num       8    352
 46    USA.DIRDES    Num       8    360
 47    USA.HERD      Num       8    368
 48    YUG.DIRDES    Num       8    376
 49    YUG.HERD      Num       8    384

                              -----Sort Information-----

                                 Sortedby:      DATE
                                 Validated:     YES
                                 Character Set: ASCII
```

Example 6.7. Remote FAME Access, using FAME CHLI

Suppose you are running FAME in a client/server environment and have FAME CHLI capability allowing you access to your FAME server. You could access your FAME remote data by specifying the port number of the tcpip service that is defined for your frdb_m and the node name of your FAME master server in your physical path. In the example below, the FAME server node name is booker, and the port number is 5555, which was designated in the FAME master command. Refer to "Starting the Master Server" in the *Guide to FAME Database Servers* for more about starting your FAME master server.

```
 /* DRIECON Database, Using FAME with REMOTE ACCESS VIA CHLI */
 /**********************************************************/

libname test1 sasefame '#5555\@booker \$FAME/util';
data a;
   set test1.driecon;
   run;

proc contents data=a;
   run;
proc means data=a n;
   run;
```

Output 6.7.1. Printout of the FAME CHLI Remote Access to FAME Data

```
                                   The SAS System                               1

                               The CONTENTS Procedure

    Data Set Name: WORK.A                                 Observations:         53
    Member Type:   DATA                                   Variables:            53
    Engine:        V8                                     Indexes:              0
    Created:       16:49 Friday, November 17, 2000        Observation Length:   424
    Last Modified: 16:49 Friday, November 17, 2000        Deleted Observations: 0
    Protection:                                           Compressed:           NO
    Data Set Type:                                        Sorted:               NO
    Label:


                     -----Engine/Host Dependent Information-----

           Data Set Page Size:         40960
           Number of Data Set Pages:   1
           First Data Page:            1
           Max Obs per Page:           96
           Obs in First Data Page:     53
           Number of Data Set Repairs: 0
           File Name:                  /tmp/SAS_work30D40000397D/a.sas7bdat
           Release Created:            8.0202M0
           Host Created:               HP-UX
           Inode Number:               3076
           Access Permission:          rw-r--r--
           Owner Name:                 myuser
           File Size (bytes):          49152


                -----Alphabetic List of Variables and Attributes-----

     # Variable  Type Len Pos Format Informat Label
    -----------------------------------------------------------------------------
     2 $N        Num    8   8                 POPULATION INCLUDING ARMED
                                              FORCES OVERSEAS (P25E)
     3 BOPMERCH  Num    8  16                 U.S. BALANCE ON MERCHANDISE
                                              TRADE (BOP)
     4 CUSA0     Num    8  24                 CPI (ALL URBAN) - ALL ITEMS
     5 CUSA0NS   Num    8  32                 CPIU - All items
     1 DATE      Num    8   0 YEAR4. 4.       Date of Observation
     6 DBTGFNS   Num    8  40
     7 DJ30C     Num    8  48                 DOW JONES: 30 INDUSTRIAL
                                              AVERAGE - CLOSE
     8 DJ65CMC   Num    8  56                 DOW JONES: 65 COMPOSITE AVERAGE
     9 FBL3Y     Num    8  64                 TREASURY BILL: SECONDARY, 3-MONTH
                                              BOND-EQUIVALENT YIELD (H15) - US
    10 FCN30YY   Num    8  72                 GOVT ISSUE: CONSTANT MATURITY,
                                              30-YR (H15) - US
    11 FIP1Q     Num    8  80                 COMMERCIAL PAPER: NON-FINAN,
                                              1-DAY QUOTED YIELD - US
    12 FIP30Q    Num    8  88                 COMMERCIAL PAPER: NON-FINAN,
                                              1-MO QUOTED YIELD - US
    13 FSCD30Y   Num    8  96                 CD: SECONDARY MKT, 1-MO YIELD - US
    14 GDP       Num    8 104                 GROSS DOMESTIC PRODUCT
    15 GDP92C    Num    8 112                 GROSS DOMESTIC PRODUCT (CHAINED)
    16 GICV      Num    8 120                 NEW CONSTRUCTION PUT IN PLACE
                                              - PUBLIC TOTAL (C30)
    17 GNP       Num    8 128                 GROSS NATIONAL PRODUCT
    18 GNP92C    Num    8 136                 GROSS NATIONAL PRODUCT
    19 HUCMPNC   Num    8 144                 HOUSING COMPLETIONS, PRIVATE
                                              - NORTH CENTRAL (C22)
    20 HUCMPNE   Num    8 152                 HOUSING COMPLETIONS, PRIVATE
                                              - NORTHEAST (C22)
    21 HUCMPSO   Num    8 160                 HOUSING COMPLETIONS,
                                              PRIVATE - SOUTH (C22)
    22 HUCMPWT   Num    8 168                 HOUSING COMPLETIONS, PRIVATE-W(C22)
```

Output 6.7.2. Proc CONTENTS of the Remote FAME Data

```
                                 The SAS System                              2

                              The CONTENTS Procedure

                -----Alphabetic List of Variables and Attributes-----

 # Variable  Type Len Pos Format Informat Label
-----------------------------------------------------------------------------
23 HUSTS     Num    8 176                 HOUSING STARTS, PRIVATE INCLUDING
                                          FARM - TOTAL (C20)
24 HUSTS1    Num    8 184                 HOUSING STARTS, PRIVATE INCL
                                          FARM - ONE UNIT (C20)
25 HUSTS1NS  Num    8 192                 HOUSING STARTS, PRIVATE INCL
                                          FARM - ONE UNIT (C20)
26 I         Num    8 200                 GROSS PRIVATE DOMESTIC INVESTMENT
27 I92C      Num    8 208                 GROSS PRIVATE DOMESTIC
                                          INVESTMENT (CHAINED)
28 ICV_G     Num    8 216                 NEW CONSTRUCTION PUT IN
                                          PLACE - TOTAL (C30)
29 JCOIN%LAG Num    8 224                 RATIO, COINCIDENT INDEX
                                          TO LAGGING INDEX (BCI)
30 JLAG      Num    8 232                 LAGGING INDICATORS COMPOSITE
                                          INDEX (BCI)
31 JLEAD     Num    8 240                 LEADING INDICATORS COMPOSITE
                                          INDEX (BCI)
32 JQIND     Num    8 248                 INDUSTRIAL PROD INDEX
                                          - TOTAL INDEX (G17)
33 JQIND20   Num    8 256                 INDUSTRIAL PROD INDEX - FOODS (G17)
35 JQIND21   Num    8 272                 INDUSTRIAL PROD INDEX -
                                          TOBACCO PRODUCTS (G17)
36 JQIND26   Num    8 280                 INDUSTRIAL PROD INDEX -
                                          PAPER & PRODUCTS (G17)
37 JQIND28   Num    8 288                 INDUSTRIAL PROD INDEX - CHEMICALS
                                          & PRODUCTS (G17)
38 JQIND37   Num    8 296                 INDUSTRIAL PROD INDEX -
                                          TRANSPORTATION EQUIPMENT (G17)
39 JQIND39   Num    8 304                 INDUSTRIAL PROD INDEX -
                                          MISC MANUFACTURES (G17)
34 JQIND208  Num    8 264                 INDUSTRIAL PROD INDEX
                                          - BEVERAGES (G17)
40 JQINDEQPB Num    8 312                 INDUSTRIAL PROD INDEX -
                                          BUSINESS EQUIPMENT (G17)
41 MNY1      Num    8 320                 MONEY SUPPLY - CURRENCY, DEMAND
                                          DEPOSITS, OTHER CHECKABLE
                                          DEPOSITS (H6)
42 MNY2      Num    8 328                 MONEY SUPPLY - M2 (H6)
43 PIDGNP    Num    8 336                 IMPLICIT PRICE DEFLATOR -
                                          GROSS NATIONAL PRODUCT
44 RUC       Num    8 344                 UNEMPLOYMENT RATE - CIVILIAN (ESIT)
45 RXC132%   Num    8 352                 EXCHANGE RATE IN NEW YORK - FRENCH
   USNS                                   FRANC PER U.S. DOLLAR (G5)
46 RXC134%   Num    8 360                 EXCHANGE RATE IN NEW YORK - GERMAN
   USNS                                   MARK PER U.S. DOLLAR (G5)
47 RXC158%   Num    8 368                 EXCHANGE RATE IN NEW YORK -
   USNS                                   JAPANESE YEN PER U.S. DOLLAR (G5)
48 RXUS%     Num    8 376                 EXCHANGE RATE IN NEW YORK - U.S.
   C112NS                                 CENTS PER BRITISH POUND (G5)
49 STR       Num    8 384                 RETAIL SALES -TOTAL (RTR)
50 WPINS     Num    8 392                 PRODUCER PRICE INDEX -
                                          ALL COMMODITIES (PPI)
51 YP        Num    8 400                 PERSONAL INCOME
52 ZA        Num    8 408                 CORPORATE PROFITS AFTER
                                          TAX EXCLUDING IVA
53 ZB        Num    8 416                 CORPORATE PROFITS BEFORE
                                          TAX EXCLUDING IVA
```

Output 6.7.3. Proc MEANS of the Remote Fame Data

```
                              The SAS System                                    4

                           The MEANS Procedure

                              Variable
                              -----------
                              DATE
                              $N
                              BOPMERCH
                              CUSA0
                              CUSA0NS
                              DBTGFNS
                              DJ30C
                              DJ65CMC
                              FBL3Y
                              FCN30YY
                              FIP1Q
                              FIP30Q
                              FSCD30Y
                              GDP
                              GDP92C
                              GICV
                              GNP
                              GNP92C
                              HUCMPNC
                              HUCMPNE
                              HUCMPSO
                              HUCMPWT
                              HUSTS
                              HUSTS1
                              HUSTS1NS
                              I
                              I92C
                              ICV_G
                              JCOIN%LAG
                              JLAG
                              JLEAD
                              JQIND
                              JQIND20
                              JQIND208
                              JQIND21
                              JQIND26
                              JQIND28
                              JQIND37
                              JQIND39
                              JQINDEQPB
                              MNY1
                              MNY2
                              PIDGNP
                              RUC
                              RXC132%USNS
                              RXC134%USNS
                              RXC158%USNS
                              RXUS%C112NS
                              STR
                              WPINS
                              YP
                              ZA
                              ZB
                              -----------
```

Output 6.7.4. Proc MEANS of the Remote FAME Data

```
                                  The SAS System                              5

                              The MEANS Procedure

Label                                                                         N
-----------------------------------------------------------------------------
Date of Observation                                                          53
POPULATION INCLUDING ARMED FORCES OVERSEAS (P25E)                            49
U.S. BALANCE ON MERCHANDISE TRADE (BOP)                                      39
CPI (ALL URBAN) - ALL ITEMS                                                  52
CPIU - All items                                                             52
                                                                             41
DOW JONES: 30 INDUSTRIAL AVERAGE - CLOSE                                     19
DOW JONES: 65 COMPOSITE AVERAGE                                              19
TREASURY BILL: SECONDARY, 3-MONTH BOND-EQUIVALENT YIELD (H15) - US           20
GOVT ISSUE: CONSTANT MATURITY, 30-YR (H15) - US                              22
COMMERCIAL PAPER: NON-FINAN, 1-DAY QUOTED YIELD - US                          1
COMMERCIAL PAPER: NON-FINAN, 1-MO QUOTED YIELD - US                          20
CD: SECONDARY MKT, 1-MO YIELD - US                                           23
GROSS DOMESTIC PRODUCT                                                       53
GROSS DOMESTIC PRODUCT (CHAINED)                                             52
NEW CONSTRUCTION PUT IN PLACE - PUBLIC TOTAL (C30)                           35
GROSS NATIONAL PRODUCT                                                       53
GROSS NATIONAL PRODUCT                                                       52
HOUSING COMPLETIONS, PRIVATE - NORTH CENTRAL (C22)                           20
HOUSING COMPLETIONS, PRIVATE - NORTHEAST (C22)                               20
HOUSING COMPLETIONS, PRIVATE - SOUTH (C22)                                   20
HOUSING COMPLETIONS, PRIVATE - WEST (C22)                                    20
HOUSING STARTS, PRIVATE INCLUDING FARM - TOTAL (C20)                         52
HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                           40
HOUSING STARTS, PRIVATE INCL FARM - ONE UNIT (C20)                           40
GROSS PRIVATE DOMESTIC INVESTMENT                                            53
GROSS PRIVATE DOMESTIC INVESTMENT (CHAINED)                                  52
NEW CONSTRUCTION PUT IN PLACE - TOTAL (C30)                                  35
RATIO, COINCIDENT INDEX TO LAGGING INDEX (BCI)                               40
LAGGING INDICATORS COMPOSITE INDEX (BCI)                                     40
LEADING INDICATORS COMPOSITE INDEX (BCI)                                     40
INDUSTRIAL PROD INDEX - TOTAL INDEX (G17)                                    52
INDUSTRIAL PROD INDEX - FOODS (G17)                                          52
INDUSTRIAL PROD INDEX - BEVERAGES (G17)                                      45
INDUSTRIAL PROD INDEX - TOBACCO PRODUCTS (G17)                               52
INDUSTRIAL PROD INDEX - PAPER & PRODUCTS (G17)                               52
INDUSTRIAL PROD INDEX - CHEMICALS & PRODUCTS (G17)                           52
INDUSTRIAL PROD INDEX - TRANSPORTATION EQUIPMENT (G17)                       52
INDUSTRIAL PROD INDEX - MISC MANUFACTURES (G17)                              52
INDUSTRIAL PROD INDEX - BUSINESS EQUIPMENT (G17)                             52
MONEY SUPPLY - CURRENCY, DEMAND DEPOSITS, OTHER CHECKABLE DEPOSITS (H6)      40
MONEY SUPPLY - M2 (H6)                                                       40
IMPLICIT PRICE DEFLATOR -  GROSS NATIONAL PRODUCT                            52
UNEMPLOYMENT RATE - CIVILIAN (ESIT)                                           9
EXCHANGE RATE IN NEW YORK - FRENCH FRANC PER U.S. DOLLAR (G5)                32
EXCHANGE RATE IN NEW YORK - GERMAN MARK PER U.S. DOLLAR (G5)                 32
EXCHANGE RATE IN NEW YORK - JAPANESE YEN PER U.S. DOLLAR (G5)                32
EXCHANGE RATE IN NEW YORK - U.S. CENTS PER BRITISH POUND (G5)                32
RETAIL SALES -TOTAL (RTR)                                                    32
PRODUCER PRICE INDEX - ALL COMMODITIES (PPI)                                 52
PERSONAL INCOME                                                              53
CORPORATE PROFITS AFTER TAX EXCLUDING IVA                                    53
CORPORATE PROFITS BEFORE TAX EXCLUDING IVA                                   53
-----------------------------------------------------------------------------
```

References

DRI/McGraw-Hill (1997), *DataLink*, Lexington, MA.

DRI/McGraw-Hill Data Search and Retrieval for Windows (1996), *DRIPRO User's Guide*, Lexington, MA.

FAME Information Services (1998), *Guide to FAME Database Servers*, 888 Seventh Avenue, 12th Floor, New York, NY 10106 [http://www.fame.com].

FAME Information Services (1995), *User's Guide to FAME*, Ann Arbor, MI.

FAME Information Services (1995), *Reference Guide to Seamless C HLI*, Ann Arbor, MI.

FAME Information Services (1995), *Command Reference for Release 7.6, Vols. 1 and 2*, Ann Arbor, MI.

Organization For Economic Cooperation and Development (1992), *Annual National Accounts: Volume I. Main Aggregates Content Documentation for Magnetic Tape Subscription*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Annual National Accounts: Volume II. Detailed Tables Technical Documentation for Magnetic Tape Subscription*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Main Economic Indicators Database Note*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Main Economic Indicators Inventory*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Main Economic Indicators OECD Statistics on Magnetic Tape Document*, Paris, France.

Organization For Economic Cooperation and Development (1992), *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Quarterly National Accounts Inventory of Series Codes*, Paris, France.

Organization For Economic Cooperation and Development (1992), *Quarterly National Accounts Technical Documentation*, Paris, France.

Acknowledgments

Many people have been instrumental in the development of the ETS Interface engine. The individuals listed here have been especially helpful.

Jeff Kaplan, FAME Information Services, Ann Arbor, MI.

Rick Langston, SAS Institute, Cary, NC.

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 7
The SASEHAVR Interface Engine (Experimental)

Chapter Contents

Chapter 7
The SASEHAVR Interface Engine (Experimental)

Overview

The SASEHAVR interface engine, experimental in Version 9, enables SAS users to read economic and financial time series data residing in a Haver Analytics Data Link Express database, and provides a seamless interface between Haver and SAS data processing.

The SASEHAVR engine uses the LIBNAME statement to enable you to specify how you would like to convert the selected time series to the same time scale. The SAS DATA step can then be used to perform further subsetting and to store the resulting time series into a SAS data set. You can perform more analysis if desired either in the same SAS session or in another session at a later time.

Getting Started

Structure of a SAS Data Set Containing Time Series Data

SAS requires time series data to be in a specific form recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASEHAVR engine provides a time ID variable called DATE.

Reading and Converting Haver DLX Time Series

The SASEHAVR engine supports reading and converting time series stored in Haver DLX databases. The SASEHAVR engine enables you to limit the range of data with the START= and END= libname options.

Using the SAS DATA Step

If desired, you can store the converted series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. Once your data is stored in a SAS data set you can use it as you would any other SAS data set.

Syntax

The SASEHAVR engine uses standard engine syntax. Options used by SASEHAVR are summarized in the table below.

Description	**Statement**	**Option**
specifies the Haver frequency.	LIBNAME *libref* SASEHAVR	FREQ=
specifies a Haver start date to limit the selection of time series that begins with the specified date.	LIBNAME *libref* SASEHAVR	START=
specifies a Haver end date to limit the selection of time series that ends with the specified date.	LIBNAME *libref* SASEHAVR	END=
specifies a list of comma delimited Haver variables to keep in the generated SAS data set.	LIBNAME *libref* SASEHAVR	KEEP=
specifies a list of comma delimited Haver variables to drop in the generated SAS data set.	LIBNAME *libref* SASEHAVR	DROP=
specifies a list of comma delimited Haver groups to keep in the generated SAS data set.	LIBNAME *libref* SASEHAVR	GROUP=
specifies a list of comma delimited Haver groups to drop in the generated SAS data set.	LIBNAME *libref* SASEHAVR	DROPGROUP=
specifies a list of comma delimited Haver sources to keep in the generated SAS data set.	LIBNAME *libref* SASEHAVR	SOURCE=
specifies a list of comma delimited Haver sources to drop in the generated SAS data set.	LIBNAME *libref* SASEHAVR	DROPSOURCE=
specifies that all selected variables should be aggregated to the frequency specified in the FREQ= option. This option is ignored if it is specified without the FREQ= option.	LIBNAME *libref* SASEHAVR	FORCE=FREQ

The LIBNAME *libref* SASEHAVR Statement

LIBNAME *libref* **sasehavr ’physical name’** *options;*

The following options can be used in the LIBNAME libref SASEHAVR statement:

FREQ=*haver_frequency*
specifies the Haver frequency. All Haver frequencies are supported by the SASEHAVR engine.

START=*start_date*
specifies the start date for the time series in the form YYYYMMDD.

END=*end_date*
specifies the end date for the time series in the form YYYYMMDD.

KEEP=*haver_variables*
specifies the list of Haver variables to be included in the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

DROP=*haver_variables*
specifies the list of Haver variables to be excluded from the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

GROUP=*haver_variables*
specifies the list of Haver groups to be included in the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

DROPGROUP=*haver_groups*
specifies the list of Haver groups to be excluded from the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

SOURCE=*haver_sources*
specifies the list of Haver sources to be included in the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

DROPSOURCE=*haver_sources*
specifies the list of Haver sources to be excluded from the generated SAS data set. this list is comma delimited and must be surrounded by quotes “”.

FORCE=FREQ
specifies that the selected variables are to be aggregated to the frequency in the FREQ= option. This option is ignored if the FREQ= option is not set.

For a more complete discussion of Haver frequencies and SAS time intervals see the section “Mapping Haver Frequencies to SAS Time Intervals” on page 223. As an example,

```
LIBNAME libref sasehavr 'physical-name'
   FREQ=MONTHLY);
```

By default, the SASEHAVR engine reads all time series in the Haver database that you reference when using your SASEHAVR libref. The *haver_startdate* is specified in the form YYYYMMDD. The start date is used to delimit the data to a specified start date.

For example, to read the time series in the TEST library starting on July 4, 1996, you would specify

```
LIBNAME test sasehavr 'physical-name'
   STARTDATE=19960704;
```

When you use the START= option, you are limiting the range of data that is read from the time series and that are converted to the desired frequency. Start dates are recommended to help you save resources when processing large databases or when processing a large number of observations. It is also possible to select specific variables to be included or excluded from the SAS data set by using the KEEP= and DROP= options.

```
LIBNAME test sasehavr 'physical-name'
   KEEP="ABC*, XYZ??";

LIBNAME test sasehavr 'physical-name'
   DROP="*SC*, #T#";
```

When the KEEP= and DROP= options are used the SAS data set that gets generated will keep or drop the variables that you select in the options. There are three wildcards currently available: '*', '?' and '#'. The '*' wildcard corresponds to any string that includes everthing in that position. The '?' means that any single alpha-numeric character is valid. And finally, the '#' wildcard corresponds to a single numeric character. You can further delimit your data by using the GROUP= and SOURCE= options and their corresponding DROPGROUP= and DROPSOURCE= options.

```
LIBNAME test sasehavr 'physical-name'
   GROUP="CBA, *ZYX";

LIBNAME test sasehavr 'physical-name'
   DROPGROUP="TKN*, XCZ?";

LIBNAME test sasehavr 'physical-name'
   SOURCE="FRB";

LIBNAME test sasehavr 'physical-name'
   DROPSOURCE="NYSE";
```

By default, SASEHAVR selects only the variables that are of the specified frequency in the FREQ= option. If this option is ignored, SASEHAVR selects the variables that match the frequency of the first selected variable. If it is desired to have all variables to be selected to have the same frequency, the FORCE=FREQ option can be specified to force the aggregation of all variables selected to be of the given frequency specified by the FREQ= option. This option is ignored if the FREQ= option is not given.

Details

The SAS Output Data Set

You can use the SAS DATA step to write the Haver converted series to a SAS data set. This allows the user the ability to easily analyze the data using SAS. You can specify the name of the output data set on the DATA statement. This causes the engine supervisor to create a SAS data set using the specified name in either the SAS WORK library, or if specified, the USER library. For more about naming your SAS data set see the section “Characteristics of SAS Data Libraries” in *SAS Language Reference: Dictionary*.

The contents of the SAS data set include the DATE of each observation, the name of each series read from the Haver database, and the label or Haver description of each series. Missing values are represented as ’.’ in the SAS data set. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL along with the SASEHAVR engine to create a view of your SAS data set.

The DATE variable in the SAS data set contains the date of the observation. The SASEHAVR engine internally maps the Haver intervals to the appropriate corresponding SAS interval.

A more detailed discussion of how to map Haver frequencies to SAS Time Intervals follows.

Mapping Haver Frequencies to SAS Time Intervals

The following table summarizes the mapping of Haver frequencies to SAS time intervals. For more information refer to “Date Intervals, Formats, and Functions” in *SAS/ETS User’s Guide*.

HAVER FREQUENCY	**SAS TIME INTERVAL**
ANNUAL	YEAR
QUARTERLY	QTR
MONTHLY	MONTH
WEEKLY (SUNDAY)	WEEK.1
WEEKLY (MONDAY)	WEEK.2
WEEKLY (TUESDAY)	WEEK.3
WEEKLY (WEDNESDAY)	WEEK.4
WEEKLY (THURSDAY)	WEEK.5
WEEKLY (FRIDAY)	WEEK.6
WEEKLY (SATURDAY)	WEEK.7
DAILY	WEEKDAY

Examples

Example 7.1. Examining the Contents of a Haver Database

To see which time series are in your HAVER database, use PROC CONTENTS with the SASEHAVR libname statement to read the contents.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=yearly start=19860101 end=19991231 force=freq;

data hwouty;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, HAVERW.DAT';
title2 'PROC CONTENTS for Time Series converted to yearly frequency';

proc contents
   data=hwouty;
run;
```

In the above example, the Haver database is called haverw and it resides in the directory referenced in lib1. The DATA statement names the SAS output data set hwouty, which will reside in saswork. All time series in the Haver haverw database are listed alphabetically in Output 7.1.1.

Output 7.1.1. Examining the Contents of Haver Analytics Database, haverw.dat

```
                       Haver Analytics Database, HAVERW.DAT
                 PROC CONTENTS for Time Series converted to yearly frequency

                              The CONTENTS Procedure

                    Alphabetic List of Variables and Attributes

 # Variable Type Len Format Label

 1 DATE     Num    8 YEAR4. Date of Observation
 2 FA       Num    8        Total Assets: All Commercial Banks (SA, Bil.$)
 3 FBAA     Num    8        Moody’s Seasoned Baa Corporate Bond Yield (% p.a.)
 4 FCDS1    Num    8        1-Month Certificates of Deposit,
                            Secondary Market (% p.a.)
 5 FDB1     Num    8        1-Month Eurodollar Deposits (London Bid) (% p.a.)
 6 FFED     Num    8        Federal Funds [Effective] Rate (% p.a.)
 7 FM1      Num    8        Money Stock: M1 (SA, Bil.$)
 8 FSLB     Num    8        Bond Buyer Index: State & Local Bonds,
                            20-Year, Genl Obligation(% p.a.)
 9 FTB3     Num    8        3-Month Treasury Bills, Auction Average (% p.a.)
10 FXEUR    Num    8        Foreign Exchange Rate: European
                            Monetary Union (US$/Euro)
11 FXFR     Num    8        Foreign Exchange Rate: France (Franc/US$)
12 FXGER    Num    8        Foreign Exchange Rate: Germany (D. Mark/US$)
13 FXJAP    Num    8        Foreign Exchange Rate: Japan  (Yen/US$)
14 FXTWB    Num    8        Nominal Broad Trade-Weighted Exchange
                            Value of the US$ (1/97=100)
15 FXTWM    Num    8        Nominal Trade-Weighted Exch Value of
                            US$ vs Major Currencies (3/73=100)
16 FXTWOTP  Num    8        Nominal Trade-Weighted Exchange
                            Value of US$ vs OITP (1/97=100)
17 FXUK     Num    8        Foreign Exchange Rate: United Kingdom (US$/Pound)
18 LICN     Num    8        Unemployment Insurance: Initial Claims,
                            State Programs (NSA, Thous)
19 LIU      Num    8        Insured Unemployment, State Programs (SA, Thous)
20 MBAMP    Num    8        MBA: Purchase Index: Mortgage Loan
                            Applications (SA, 3/16/90=100)
21 MBAMR    Num    8        MBA: Refinancing Index: Mortgage Loan
                            Applications (SA, 3/16/90=100)
```

You could use the following SAS statements to create a SAS data set named hwouty and to print its contents.

```
libname lib1 sasehavr ’path-to-haver-data’
             freq=yearly start=19860101 end=19991231 force=freq;

data hwouty;
   set lib1.haverw;
run;

title1 ’Haver Analytics Database, Frequency=yearly, infile=haverw.dat’;

proc print
   data=hwouty;
run;
```

The libref above specifies that all time series in the haverw database be converted to yearly frequency but to only select the range of data from January 1, 1986, to December 31, 1999. The resulting SAS data set hwouty is shown in Output 7.1.2.

Output 7.1.2. Defining a Range Inside the Data Range Using the START=19860101 END=19991231 Libname Options

```
          Haver Analytics Database, Frequency=yearly, infile=haverw.dat

Obs   DATE     FA       FBAA     FCDS1      FDB1      FFED       FM1       FSLB

  1   1986   2703.9   10.4008   6.61192   6.80077   6.83358    666.19   7.33635
  2   1987   2760.2   10.5667   6.75192   6.87827   6.65942    743.75   7.64792
  3   1988   2941.1   10.8379   7.59189   7.69434   7.55558    774.68   7.68192
  4   1989   3148.7   10.1783   9.11365   9.16538   9.22519    782.21   7.22615
  5   1990   3299.6   10.3542   8.15481   8.15885   8.10923    810.86   7.27346
  6   1991   3411.2    9.8102   5.83769   5.83192   5.73269    859.52   6.91904
  7   1992   3466.0    8.9779   3.64981   3.63654   3.53415    965.73   6.43698
  8   1993   3626.4    7.9381   3.11547   3.07264   3.02154   1078.16   5.58923
  9   1994   3879.9    8.6288   4.38788   4.35115   4.19154   1145.36   6.18635
 10   1995   4218.1    8.1994   5.86596   5.85904   5.83865   1143.26   5.94885
 11   1996   4414.8    8.0542   5.34827   5.31846   5.29769   1106.75   5.75519
 12   1997   4836.6    7.8710   5.54038   5.51615   5.45792   1069.85   5.52269
 13   1998   5268.4    7.2206   5.49269   5.45365   5.35500   1080.61   5.08755
 14   1999   5613.7    7.8670   5.19887   5.15038   4.97231   1101.94   5.43365

Obs     FTB3       FXEUR       FXFR       FXGER      FXJAP      FXTWB      FXTWM

  1   5.97673       .        6.93509    2.17534    168.812       .       108.007
  2   5.81765       .        6.02454    1.80222    145.063       .        95.671
  3   6.68500       .        5.94978    1.75406    128.065       .        88.710
  4   8.11654       .        6.37848    1.88022    138.052       .        92.390
  5   7.51096       .        5.44622    1.61630    144.857       .        88.394
  6   5.42077       .        5.64333    1.65997    134.593       .        86.949
  7   3.45415       .        5.28946    1.56055    126.726       .        85.362
  8   3.01654       .        5.66371    1.65344    111.331       .        87.750
  9   4.28673       .        5.54375    1.62090    102.159       .        86.229
 10   5.51058       .        4.98848    1.43280     94.058     92.516     81.393
 11   5.02096       .        5.11492    1.50456    108.756     97.416     85.214
 12   5.06885       .        5.82993    1.73197    120.930    104.342     91.767
 13   4.80755       .        5.90226    1.76062    131.056    116.260     96.540
 14   4.66269    1.06529     6.15538    1.83535    113.723    116.458     94.434

Obs   FXTWOTP      FXUK        LICN        LIU       MBAMP      MBAMR

  1      .       1.46695    376.227    2631.54       .           .
  2      .       1.63477    326.156    2273.13       .           .
  3      .       1.78245    312.881    2081.45       .           .
  4      .       1.63838    328.813    2177.08       .           .
  5      .       1.78467    387.002    2539.94     87.415     100.49
  6      .       1.76851    447.600    3341.83    107.627     350.35
  7      .       1.76944    407.340    3206.81    131.031     726.22
  8      .       1.50187    344.934    2766.17    157.438    1077.45
  9      .       1.53227    340.054    2677.42    142.627     271.09
 10    92.513    1.57826    357.038    2592.88    165.213     335.96
 11    98.232    1.56057    351.358    2549.88    183.800     448.80
 12   104.509    1.63838    321.513    2298.85    205.631     525.54
 13   125.636    1.65717    317.077    2215.15    265.869    1742.97
 14   129.514    1.61833    301.570    2187.38    276.292     799.40
```

Example 7.2. Viewing Quarterly Time Series from a Haver Database

Consider the following statements for quarterly frequency conversion of all time series for the period spanning April 1, 1996, to December 31, 1999.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=quarterly start=19960401 end=19991231 force=freq;

data hwoutq;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=quarterly, infile=haverw.dat';

proc print
   data=hwoutq;
run;
```

The resulting SAS data set hwoutq is shown in Output 7.2.1.

Output 7.2.1. Defining a Range Inside the Data Range Using the START=19960401 END=19991231 Libname Options

```
       Haver Analytics Database, Frequency=quarterly, infile=haverw.dat

Obs    DATE     FA      FBAA     FCDS1     FDB1     FFED     FM1      FSLB

  1  1996Q2  4294.9  8.29462  5.33923  5.31692  5.25231  1119.21  5.98000
  2  1996Q3  4358.4  8.29385  5.35385  5.31462  5.29077  1104.83  5.84385
  3  1996Q4  4414.8  7.92538  5.35769  5.33077  5.29154  1082.02  5.65538
  4  1997Q1  4532.1  8.06538  5.36615  5.33154  5.24231  1076.49  5.70385
  5  1997Q2  4607.0  8.19615  5.57308  5.56000  5.52692  1065.30  5.70077
  6  1997Q3  4674.1  7.76923  5.54769  5.53923  5.54692  1069.02  5.37846
  7  1997Q4  4836.6  7.45308  5.67462  5.63385  5.51143  1068.60  5.30769
  8  1998Q1  4949.4  7.25615  5.55769  5.52615  5.51667  1076.71  5.12000
  9  1998Q2  5001.2  7.25846  5.56308  5.53077  5.48231  1078.50  5.18385
 10  1998Q3  5186.7  7.12846  5.55308  5.49923  5.54857  1076.00  5.07923
 11  1998Q4  5268.4  7.23923  5.29692  5.25846  4.87000  1091.25  4.97571
 12  1999Q1  5254.7  7.39308  4.90769  4.85692  4.72615  1097.24  5.04583
 13  1999Q2  5334.6  7.71923  4.88769  4.82462  4.74769  1103.06  5.19769
 14  1999Q3  5398.7  8.09077  5.23077  5.18154  5.09077  1098.12  5.54214
 15  1999Q4  5613.7  8.23643  5.72857  5.69643  5.32462  1109.35  5.91077

Obs    FTB3      FXEUR       FXFR      FXGER      FXJAP      FXTWB      FXTWM

  1  5.03615     .         5.15645    1.52185    107.474     97.275    85.3792
  2  5.13231     .         5.09276    1.49715    108.948     97.553    85.2077
  3  4.97231     .         5.17515    1.53074    112.800     98.381    85.7977
  4  5.06077     .         5.57572    1.65240    120.876    101.363    89.8692
  5  5.07769     .         5.77722    1.71354    119.903    102.648    91.3038
  6  5.06000     .         6.08371    1.80610    117.875    104.578    92.4538
  7  5.07692     .         5.88307    1.75583    125.067    108.778    93.4400
  8  5.07385     .         6.08678    1.81652    128.173    115.081    95.8431
  9  5.00231     .         6.01921    1.79532    135.492    115.607    97.2685
 10  4.88692     .         5.92476    1.76694    140.123    119.040    99.1800
 11  4.30571     .         5.57831    1.66371    120.437    115.312    93.8700
 12  4.42333    1.12328    5.82563    1.73710    116.403    116.150    93.8877
 13  4.44615    1.05832    6.19949    1.84847    120.732    117.200    96.0162
 14  4.69000    1.04788    6.26160    1.86699    114.008    116.676    95.1715
 15  5.07077    1.03824    6.32198    1.88501    104.461    115.851    92.7857

Obs  FXTWOTP      FXUK       LICN        LIU       MBAMP      MBAMR

  1   97.634    1.52383    310.154    2574.23    183.831     280.26
  2   98.569    1.55393    298.415    2498.15    183.200     247.07
  3   99.619    1.63358    360.369    2480.23    191.477     396.30
  4  100.418    1.63421    381.885    2385.00    190.592     402.50
  5  101.251    1.63501    290.623    2286.08    193.462     322.79
  6  103.986    1.62640    286.992    2274.69    214.515     574.10
  7  112.382    1.65792    326.554    2249.62    223.954     802.78
  8  123.847    1.64591    375.500    2213.77    245.038    1828.42
  9  122.736    1.65408    284.869    2153.69    259.946    1235.27
 10  128.065    1.65148    284.269    2266.46    269.354    1602.43
 11  127.895    1.67722    323.669    2226.69    289.138    2305.75
 12  130.027    1.63638    371.100    2219.46    259.885    1457.69
 13  128.897    1.60778    268.554    2192.54    290.262     914.37
 14  129.076    1.59788    262.931    2204.54    276.808     457.92
 15  130.016    1.63035    303.543    2136.86    278.079     398.46
```

Example 7.3. Viewing Monthly Time Series from a Haver Database

Suppose you wish to convert your time series to the monthly frequency. An example of monthly conversion would look like this:

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=monthly start=19990401 end=19991231 force=freq;

data hwoutm;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=monthly, infile=haverw.dat';

proc print
   data=hwoutm;
run;
```

The result from using the range of April 1, 1999, to December 31, 1999, is shown in Output 7.3.1.

Output 7.3.1. Defining a Range Inside the Data Range Using the START=19960401 END=19991231 Libname Options

```
         Haver Analytics Database, Frequency=monthly, infile=haverw.dat

Obs      DATE      FA      FBAA     FCDS1    FDB1     FFED      FM1      FSLB

 1     APR1999   5288.0   7.4860   4.848   4.8020   4.7200   1107.38   5.0760
 2     MAY1999   5290.7   7.7225   4.845   4.7825   4.7725   1102.48   5.1825
 3     JUN1999   5334.6   8.0075   4.980   4.8950   4.7500   1099.48   5.3650
 4     JUL1999   5335.9   7.9600   5.136   5.0740   4.9850   1099.40   5.3620
 5     AUG1999   5385.0   8.1500   5.245   5.1925   5.0175   1099.08   5.5800
 6     SEP1999   5398.7   8.1950   5.335   5.3050   5.2340   1095.63   5.6920
 7     OCT1999   5463.6   8.3560   5.360   5.3200   5.2000   1100.45   5.9150
 8     NOV1999   5552.8   8.1450   5.450   5.3975   5.3575   1108.74   5.8550
 9     DEC1999   5613.7   8.1900   6.320   6.3120   5.3980   1119.00   5.9520

Obs     FTB3      FXEUR       FXFR       FXGER      FXJAP      FXTWB      FXTWM

 1     4.2820    1.07066    6.12704    1.82686    119.732    117.206    95.7860
 2     4.5075    1.06300    6.17138    1.84010    122.000    116.913    95.7950
 3     4.5900    1.03823    6.31818    1.88385    120.713    117.480    96.5250
 4     4.6000    1.03606    6.33330    1.88836    119.650    117.490    96.8240
 5     4.7550    1.06150    6.18013    1.84268    113.530    116.465    94.7725
 6     4.7280    1.04903    6.25345    1.86460    107.435    115.870    93.5050
 7     4.8750    1.06840    6.14034    1.83084    106.066    115.428    92.2960
 8     5.0650    1.03493    6.33883    1.89005    104.893    116.045    92.8325
 9     5.2320    1.01074    6.49014    1.93514    102.512    116.118    93.2380

Obs    FXTWOTP      FXUK       LICN        LIU       MBAMP      MBAMR

 1     129.322    1.60936    278.620    2185.60    277.200    1120.42
 2     128.555    1.61543    259.000    2188.75    288.275     898.78
 3     128.708    1.59815    265.525    2205.00    308.575     672.40
 4     128.212    1.57492    316.900    2220.60    285.860     513.52
 5     129.233    1.60693    235.475    2201.00    277.450     452.08
 6     130.000    1.61755    222.925    2188.00    264.850     394.28
 7     130.728    1.65536    253.720    2148.20    272.380     404.44
 8     129.903    1.62235    284.300    2132.25    295.475     446.65
 9     129.394    1.61174    368.760    2129.20    269.860     353.92
```

Example 7.4. Viewing Weekly Time Series from a Haver Database

An example of weekly data spanning September 1, 1999, to December 31, 1999, is shown in Output 7.4.1.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=weekly start=19990901 end=19991231;

data hwoutw;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=weekly, infile=haverw.dat';

proc print
   data=hwoutw;
run;
```

Output 7.4.1. Defining a Range Inside the Data Range Using the START=19960401 END=19991231 Libname Options

```
          Haver Analytics Database, Frequency=weekly, infile=haverw.dat

Obs      DATE    FA    FBAA FCDS1 FDB1 FFED   FM1   FSLB FTB3  FXEUR  FXFR

  1 30AUG1999 5365.4 8.21  5.32 5.26 5.34 1095.5 5.67 4.88 1.0580 6.2001
  2 06SEP1999 5378.4 8.20  5.34 5.33 5.16 1096.7 5.66 4.72 1.0529 6.2303
  3 13SEP1999 5393.6 8.18  5.34 5.31 5.24 1094.7 5.69 4.66 1.0402 6.3061
  4 20SEP1999 5413.5 8.19  5.34 5.32 5.16 1097.5 5.71 4.66 1.0450 6.2773
  5 27SEP1999 5398.7 8.24  5.35 5.31 5.27 1093.6 5.73 4.72 1.0583 6.1989
  6 04OCT1999 5415.3 8.28  5.37 5.33 5.27 1101.7 5.80 4.73 1.0690 6.1361
  7 11OCT1999 5415.8 8.40  5.36 5.33 5.17 1097.9 5.89 4.78 1.0795 6.0765
  8 18OCT1999 5434.5 8.44  5.36 5.32 5.18 1100.6 5.98 4.99 1.0789 6.0799
  9 25OCT1999 5463.6 8.42  5.36 5.31 5.18 1101.6 5.99 5.00 1.0563 6.2103
 10 01NOV1999 5438.9 8.27  5.36 5.31 5.27 1107.2 5.88 5.00 1.0465 6.2682
 11 08NOV1999 5459.1 8.13  5.37 5.33 5.20 1105.4 5.83 5.03 1.0379 6.3203
 12 15NOV1999 5472.7 8.06  5.51 5.45 5.44 1109.4 5.84 5.12 1.0328 6.3514
 13 22NOV1999 5552.8 8.12  5.56 5.50 5.52 1112.3 5.87 5.11 1.0225 6.4154
 14 29NOV1999 5554.4 8.17  6.24 6.38 5.63 1109.4 5.91 5.20 1.0055 6.5237
 15 06DEC1999 5573.4 8.08  6.39 6.36 5.45 1110.8 5.89 5.05 1.0211 6.4240
 16 13DEC1999 5603.0 8.17  6.44 6.38 5.44 1110.1 5.96 5.21 1.0102 6.4934
 17 20DEC1999 5638.3 8.29  6.48 6.44 5.46 1120.2 6.00 5.40 1.0100 6.4946
 18 27DEC1999 5613.7 8.24  6.05 6.00 5.01 1134.9 6.00 5.30 1.0069 6.5150

Obs  FXGER   FXJAP   FXTWB  FXTWM  FXTWOTP   FXUK    LICN   LIU  MBAMP  MBAMR

  1 1.8487  109.82  116.13  94.09   129.61  1.6021  235.8  2219  254.1  395.9
  2 1.8577  109.73  116.19  93.93   130.04  1.6189  204.3  2165  278.7  394.1
  3 1.8803  105.47  115.63  93.20   129.94  1.6151  219.1  2180  266.4  398.7
  4 1.8717  104.72  115.53  92.80   130.41  1.6341  232.5  2188  260.2  388.4
  5 1.8483  106.18  115.48  92.56   130.74  1.6474  246.4  2181  288.8  414.6
  6 1.8296  107.18  115.29  92.46   130.43  1.6536  278.9  2176  262.7  412.2
  7 1.8118  106.53  115.30  92.22   130.90  1.6582  234.6  2117  277.3  396.3
  8 1.8128  105.67  115.19  92.15   130.75  1.6684  250.9  2138  255.4  383.2
  9 1.8517  104.77  115.88  92.09   130.82  1.6492  257.8  2129  277.7  415.9
 10 1.8690  104.83  115.87  92.39   130.23  1.6374  297.1  2165  268.8  443.7
 11 1.8845  105.14  116.07  92.85   129.93  1.6185  262.6  2052  298.5  443.1
 12 1.8938  105.63  116.12  93.04   129.74  1.6187  309.2  2187  315.7  481.8
 13 1.9129  103.97  116.12  93.05   129.71  1.6148  268.3  2125  298.9  418.0
 14 1.9452  102.37  116.60  93.53   130.10  1.5981  378.7  2105  287.9  386.4
 15 1.9154  102.69  116.02  92.96   129.62  1.6235  318.2  2098  285.8  388.1
 16 1.9361  103.32  116.28  93.59   129.19  1.6124  329.6  2150  284.3  354.1
 17 1.9365  102.01  115.90  93.24   128.84  1.6090  377.7  2157  253.3  338.1
 18 1.9425  102.17  115.79  92.87   129.22  1.6157  439.6  2136  238.0  302.9
```

Example 7.5. Viewing Daily Time Series from a Haver Database

Consider viewing the Haver Analytics daily database named haverd. The contents of this database can be seen by submitting the following data step.

```
libname lib1 sasehavr 'path-to-haver-data'
             freq=daily start=19991201 end=19991231;

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, HAVERD.DAT';
title2 'PROC CONTENTS for Time Series converted to daily frequency';

proc contents
   data=hwoutd;
run;
```

Output 7.5.1 shows the output of PROC CONTENTS with the following time series.

Output 7.5.1. Examining the Contents of a Daily Haver Analytics Database, haverd.dat

```
                       Haver Analytics Database, HAVERD.DAT
            PROC CONTENTS for Time Series converted to daily frequency

                              The CONTENTS Procedure

                    Alphabetic List of Variables and Attributes

# Variable Type Len Format Label

1 DATE     Num    8 DATE9. Date of Observation
2 FAAA     Num    8        Moody's Seasoned Aaa Corporate Bond Yield (% p.a.)
3 FBAA     Num    8        Moody's Seasoned Baa Corporate Bond Yield (% p.a.)
4 GSCITR   Num    8        Goldman Sachs Commodity Total
                           Return Index (12/31/69=100)
5 PFALL    Num    8        KR-CRB Futures Price Index:
                           All Commodities (1967=100)
6 PFGR     Num    8        KR-CRB Futures Price Index: Grains (1967=100)
7 SP500    Num    8        Standard & Poor's 500 Stock Price Index (1941-43=10)
8 SPDJC    Num    8        Stock Price Averages: Dow Jones
                           65 Composite, NYSE (Close)
```

Example 7.6. Limiting the Range of Time Series from a Haver Database

Suppose we limit the range of data to the month of December:

```
libname lib1 sasehavr 'path-to-haver-data'
                freq=daily start=19991201 end=19991231;

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';

proc print
   data=hwoutd;
run;
```

Note that Output 7.6.1 for daily conversion shows the frequency as the SAS time interval for WEEKDAY.

Output 7.6.1. Defining a Range Inside the Data Range Using the START=19991201 END=19991231 Libname Options

```
          Haver Analytics Database, Frequency=daily, infile=haverd.dat

 Obs         DATE    FAAA    FBAA    GSCITR    PFALL    PFGR     SP500     SPDJC

   1    01DEC1999    7.50    8.18   2676.57   203.98   158.44   1397.72   3097.77
   2    02DEC1999    7.53    8.19   2726.32   204.27   157.35   1409.04   3109.94
   3    03DEC1999    7.46    8.13   2720.04   204.39   155.77   1433.30   3165.46
   4    06DEC1999    7.45    8.12   2759.68   204.78   158.01   1423.33   3146.24
   5    07DEC1999    7.42    8.08   2738.25   204.75   157.71   1409.17   3111.07
   6    08DEC1999    7.44    8.10   2751.78   203.91   155.42   1403.88   3099.17
   7    09DEC1999    7.43    8.09   2718.13   202.47   152.93   1408.11   3101.33
   8    10DEC1999    7.37    8.03   2684.52   202.35   153.56   1417.04   3125.78
   9    13DEC1999    7.40    8.06   2694.15   201.64   151.02   1415.22   3117.77
  10    14DEC1999    7.50    8.16   2728.49   202.57   152.70   1403.17   3113.72
  11    15DEC1999    7.53    8.18   2755.69   203.69   152.69   1413.32   3133.11
  12    16DEC1999    7.59    8.24   2801.98   205.21   153.27   1418.78   3134.18
  13    17DEC1999    7.59    8.23   2810.22   205.51   154.24   1421.03   3142.46
  14    20DEC1999    7.63    8.27   2810.85   206.13   156.06   1418.09   3114.00
  15    21DEC1999    7.66    8.30   2793.80   203.88   155.01   1433.43   3120.48
  16    22DEC1999    7.66    8.28   2755.95   203.51   156.61   1436.13   3122.06
  17    23DEC1999    7.67    8.29   2776.63   204.66   157.45   1458.34   3175.80
  18    24DEC1999     .       .         .        .        .         .         .
  19    27DEC1999    7.65    8.26   2787.82   204.04   155.96   1457.10   3183.50
  20    28DEC1999    7.66    8.28   2811.95   204.10   156.14   1457.66   3203.45
  21    29DEC1999    7.63    8.24   2808.26   205.10   155.42   1463.46   3201.93
  22    30DEC1999    7.61    8.23   2769.59   205.14   156.64   1464.47   3203.93
  23    31DEC1999    7.64    8.18   2770.01      .        .      1469.25   3214.38
```

Example 7.7. Using the WHERE Statement to Subset Time Series from a Haver Database

Using a WHERE statement in the DATA step can be useful for further subsetting.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=daily start=19991101 end=19991231;

data hwoutd;
   set lib1.haverd;
   where date  between '01nov99'd and '01dec99'd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
proc print
   data=hwoutd;
run;
```

Output 7.7.1 shows that the time slice of November 1, 1999, to December 31, 1999, is narrowed further by the DATE test on the WHERE statement to stop at December 1, 1999.

Output 7.7.1. Defining a Range Using START=19991101 END=19991231 Along with the WHERE statement

```
          Haver Analytics Database, Frequency=daily, infile=haverd.dat

Obs        DATE    FAAA    FBAA    GSCITR    PFALL     PFGR     SP500     SPDJC

  1   01NOV1999    7.42    8.33   2586.58   203.52   160.97   1354.12   3096.19
  2   02NOV1999    7.37    8.30   2575.52   202.70   162.19   1347.74   3084.80
  3   03NOV1999    7.35    8.28   2596.41   203.53   163.38   1354.93   3091.45
  4   04NOV1999    7.29    8.23   2620.53   203.97   163.96   1362.64   3091.00
  5   05NOV1999    7.25    8.19   2613.22   203.02   162.70   1370.23   3108.64
  6   08NOV1999    7.26    8.20   2616.61   203.14   163.57   1377.01   3114.84
  7   09NOV1999    7.28    8.13   2659.89   204.41   161.94   1365.28   3083.00
  8   10NOV1999    7.31    8.13   2684.81   204.84   158.22   1373.46   3079.00
  9   11NOV1999    7.32    8.12   2670.60   204.17   158.00   1381.46   3083.25
 10   12NOV1999    7.28    8.06   2709.32   205.63   156.76   1396.06   3132.60
 11   15NOV1999    7.28    8.03   2726.36   206.66   158.93   1394.39   3122.17
 12   16NOV1999    7.30    8.01   2738.96   206.32   159.30   1420.07   3160.55
 13   17NOV1999    7.36    8.08   2761.09   205.65   157.39   1410.71   3129.24
 14   18NOV1999    7.39    8.10   2712.71   202.74   155.10   1424.94   3156.36
 15   19NOV1999    7.38    8.09   2749.90   202.75   155.68   1422.00   3138.01
 16   22NOV1999    7.40    8.12   2784.27   203.14   156.96   1420.94   3139.59
 17   23NOV1999    7.40    8.12   2758.29   203.05   157.13   1404.64   3119.44
 18   24NOV1999    7.41    8.11   2776.49   202.76   155.82   1417.08   3113.18
 19   25NOV1999     .       .         .        .        .         .         .
 20   26NOV1999    7.44    8.13   2772.63   202.79   154.40   1416.62   3103.57
 21   29NOV1999    7.52    8.20   2728.93   203.14   153.37   1407.83   3086.44
 22   30NOV1999    7.50    8.17   2657.75   204.07   157.39   1388.91   3083.49
 23   01DEC1999    7.50    8.18   2676.57   203.98   158.44   1397.72   3097.77
```

Example 7.8. Using the KEEP Option to Subset Time Series from a Haver Database

To select specific time series the KEEP or DROP option may also be used as follows.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=daily start=19991101 end=19991231 keep="SP*";

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
proc print
   data=hwoutd;
run;
```

Output 7.8.1 shows two series that are selected by using KEEP="SP*" on the libname statement.

Output 7.8.1. Using the KEEP option along with defining a Range Using START=19991101 END=19991231

```
           Haver Analytics Database, Frequency=daily, infile=haverd.dat

                      Obs         DATE     SP500      SPDJC

                        1    01NOV1999    1354.12    3096.19
                        2    02NOV1999    1347.74    3084.80
                        3    03NOV1999    1354.93    3091.45
                        4    04NOV1999    1362.64    3091.00
                        5    05NOV1999    1370.23    3108.64
                        6    08NOV1999    1377.01    3114.84
                        7    09NOV1999    1365.28    3083.00
                        8    10NOV1999    1373.46    3079.00
                        9    11NOV1999    1381.46    3083.25
                       10    12NOV1999    1396.06    3132.60
                       11    15NOV1999    1394.39    3122.17
                       12    16NOV1999    1420.07    3160.55
                       13    17NOV1999    1410.71    3129.24
                       14    18NOV1999    1424.94    3156.36
                       15    19NOV1999    1422.00    3138.01
                       16    22NOV1999    1420.94    3139.59
                       17    23NOV1999    1404.64    3119.44
                       18    24NOV1999    1417.08    3113.18
                       19    25NOV1999          .          .
                       20    26NOV1999    1416.62    3103.57
                       21    29NOV1999    1407.83    3086.44
                       22    30NOV1999    1388.91    3083.49
                       23    01DEC1999    1397.72    3097.77
                       24    02DEC1999    1409.04    3109.94
                       25    03DEC1999    1433.30    3165.46
                       26    06DEC1999    1423.33    3146.24
                       27    07DEC1999    1409.17    3111.07
                       28    08DEC1999    1403.88    3099.17
                       29    09DEC1999    1408.11    3101.33
                       30    10DEC1999    1417.04    3125.78
                       31    13DEC1999    1415.22    3117.77
                       32    14DEC1999    1403.17    3113.72
                       33    15DEC1999    1413.32    3133.11
                       34    16DEC1999    1418.78    3134.18
                       35    17DEC1999    1421.03    3142.46
                       36    20DEC1999    1418.09    3114.00
                       37    21DEC1999    1433.43    3120.48
                       38    22DEC1999    1436.13    3122.06
                       39    23DEC1999    1458.34    3175.80
                       40    24DEC1999          .          .
                       41    27DEC1999    1457.10    3183.50
                       42    28DEC1999    1457.66    3203.45
                       43    29DEC1999    1463.46    3201.93
                       44    30DEC1999    1464.47    3203.93
                       45    31DEC1999    1469.25    3214.38
```

The DROP option can be used to drop specific variables from a Haver dataset. To specify this option, use DROP= instead of KEEP= as shown above.

Example 7.9. Using the SOURCE Option to Subset Time Series from a Haver Database

To select specific variables that belong to a certain source, the SOURCE or DROPSOURCE option may also be used much like KEEP and DROP.

```
libname lib1 sasehavr 'path-to-haver-data'
            freq=daily start=19991101 end=19991223 source="FRB";

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
proc print
   data=hwoutd;
run;
```

Output 7.9.1 shows two series that are selected by using SOURCE="FRB" on the libname statement.

Output 7.9.1. Using the SOURCE option along with defining a Range Using START=19991101 END=19991213

```
          Haver Analytics Database, Frequency=daily, infile=haverd.dat

                    Obs          DATE     FAAA     FBAA

                      1     01NOV1999     7.42     8.33
                      2     02NOV1999     7.37     8.30
                      3     03NOV1999     7.35     8.28
                      4     04NOV1999     7.29     8.23
                      5     05NOV1999     7.25     8.19
                      6     08NOV1999     7.26     8.20
                      7     09NOV1999     7.28     8.13
                      8     10NOV1999     7.31     8.13
                      9     11NOV1999     7.32     8.12
                     10     12NOV1999     7.28     8.06
                     11     15NOV1999     7.28     8.03
                     12     16NOV1999     7.30     8.01
                     13     17NOV1999     7.36     8.08
                     14     18NOV1999     7.39     8.10
                     15     19NOV1999     7.38     8.09
                     16     22NOV1999     7.40     8.12
                     17     23NOV1999     7.40     8.12
                     18     24NOV1999     7.41     8.11
                     19     25NOV1999      .        .
                     20     26NOV1999     7.44     8.13
                     21     29NOV1999     7.52     8.20
                     22     30NOV1999     7.50     8.17
                     23     01DEC1999     7.50     8.18
                     24     02DEC1999     7.53     8.19
                     25     03DEC1999     7.46     8.13
                     26     06DEC1999     7.45     8.12
                     27     07DEC1999     7.42     8.08
                     28     08DEC1999     7.44     8.10
                     29     09DEC1999     7.43     8.09
                     30     10DEC1999     7.37     8.03
                     31     13DEC1999     7.40     8.06
```

Example 7.10. Using the GROUP Option to Subset Time Series from a Haver Database

To select specific variables that belong to a certain group, the GROUP or DROPGROUP option may also be used much like KEEP and DROP.

```
libname lib1 sasehavr 'path-to-haver-date'
            freq=week.6 start=20000107 end=20001007 group="C*";

data hwoutw;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=week.6, infile=haverw.dat';
proc print
   data=hwoutw;
run;
```

Output 7.10.1 shows two series that are selected by using GROUP="C*" on the libname statement.

Output 7.10.1. Using the GROUP option along with defining a Range Using START=20000107 END=20000609

```
       Haver Analytics Database, Frequency=week.6, infile=haverw.dat

                    Obs         DATE    MBAMP    MBAMR

                      1    07JAN2000    254.6    350.8
                      2    14JAN2000    292.0    390.5
                      3    21JAN2000    286.1    413.6
                      4    28JAN2000    292.6    384.4
                      5    04FEB2000    307.1    436.7
                      6    11FEB2000    270.8    373.1
                      7    18FEB2000    291.1    372.9
                      8    25FEB2000    278.4    346.6
                      9    03MAR2000    291.7    377.5
                     10    10MAR2000    290.0    361.8
                     11    17MAR2000    293.5    346.2
                     12    24MAR2000    312.2    386.6
                     13    31MAR2000    293.5    340.6
                     14    07APR2000    316.6    364.2
                     15    14APR2000    300.8    354.8
                     16    21APR2000    302.8    341.9
                     17    28APR2000    299.4    336.2
                     18    05MAY2000    322.4    331.3
                     19    12MAY2000    296.6    330.9
                     20    19MAY2000    326.3    328.9
                     21    26MAY2000    302.8    294.4
                     22    02JUN2000    335.0    318.1
                     23    09JUN2000    309.5    329.4
```

Reference

Haver Analytics (2001), *DLX API Programmer's Reference*, New York, NY.

Acknowledgments

Many people have been instrumental in the development of the ETS Interface engine. The individuals listed here have been especially helpful.

Maurine Haver, Haver Analytics, New York, NY.

Lai Cheng, Haver Analytics, New York, NY.

Rick Langston, SAS Institute, Cary, NC.

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 8
Using the Output Delivery System

Chapter Contents

Chapter 8
Using the Output Delivery System

Overview

In the latest version of SAS software, all SAS/ETS procedures use the Output Delivery System (ODS) to manage their output. This includes managing the form in which the output appears as well as its organization and format. The default for SAS/ETS procedures is to produce the usual SAS listing file. However, by using the features of the Output Delivery System, you can make changes to the format and appearance of your SAS output. In particular, you can

- display your output in hypertext markup language (HTML).
- display your output in Rich-Text-Format (RTF).
- create SAS data sets directly from output tables.
- select or exclude individual output tables.
- customize the layout, format, and headers of your output.

ODS features can provide you with a powerful tool for managing your output. This chapter provides background material and illustrates typical applications of ODS with SAS/ETS software.

For complete documentation on the Output Delivery System, refer to *SAS Output Delivery System User's Guide*.

Output Objects and ODS Destinations

All SAS procedures produce *output objects* that the Output Delivery System delivers to various *ODS destinations*, according to the default specifications for the procedure or to your own specifications.

All output objects (for example, a table of parameter estimates) consist of two component parts:

- the data component, which consists of the results computed by a SAS procedure.
- the template, which contains rules for formatting and displaying the results.

When you invoke a SAS procedure, the procedure sends all output to the Output Delivery System. ODS then routes the output to all open destinations. You define the form the output should take when you specify an ODS destination. Supported destinations are as follows:

- Listing destination (the standard SAS listing), which is the default.
- HTML destination, hypertext markup language.
- Output destination, SAS data set.

Future versions of ODS will support the following additional destinations:

- the ODS Output Document for modifying and replaying output without rerunning the procedure that created it.
- Rich Text Format (RTF) for inclusion in Microsoft Word.
- postscript and PCL for high fidelity printers.

You can activate multiple ODS destinations at the same time, so that a single procedure step can route output to multiple destinations. If you do not supply any ODS statements, ODS delivers all output to the SAS listing, which is the default.

Each output object has an associated template that defines its presentation format. You can modify the presentation of the output by using the TEMPLATE procedure to alter these templates or to create new templates. You can also specify stylistic elements for ODS destinations, such as cell formats and headers, column ordering, colors, and fonts. For detailed information, refer to the chapter titled "The Template Procedure" in the *SAS Procedures Guide*.

Using the Output Delivery System

The ODS statement is a global statement that enables you to provide instructions to the Output Delivery System. You can use ODS statements to specify options for different ODS destinations, select templates to format your output, and select and exclude output. You can also display the names of individual output tables as they are generated.

In order to select, exclude, or modify a table, you must first know its name. You can obtain the table names in several ways:

- For any SAS/ETS procedure, you can obtain table names from the individual procedure chapter or from the individual procedure section of the SAS online Help system.
- For any SAS procedure, you can use the SAS Explorer window to view the names of the tables created in your SAS run (see the section "Using ODS with the SAS Explorer" on page 247 for more information).
- For any SAS procedure, you can use the ODS TRACE statement to find the names of tables created in your SAS run. The ODS TRACE statement writes identifying information to the SAS log (or, optionally, to the SAS listing) for each generated output table.

Specify the ODS TRACE ON statement prior to the procedure statements that create the output for which you want information. For example, the following statements write the trace record for the specific tables created in this AUTOREG procedure step.

```
ods trace on;
proc autoreg;
   model y1 = time;
   model y2 = time;
run;
```

By default, the trace record is written to the SAS log, as displayed in Figure 8.1. Alternatively, you can specify the LISTING option, which writes the information, interleaved with the procedure output, to the SAS listing (see Example 8.1).

```
   ods trace on;
   proc autoreg;
      model y1 = time;
      model y2 = time;
   run;

.
.
.

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   ets.autoreg.ParameterEstimates
Path:       Autoreg.Model1.OLSEst.ParameterEstimates
-------------

.
.
.

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   ets.autoreg.ParameterEstimates
Path:       Autoreg.Model2.OLSEst.ParameterEstimates
-------------
```

Figure 8.1. Partial Contents of the SAS Log: Result of the ODS TRACE Statement

Figure 8.1 displays the trace record, which contains the name of each created table and its associated label, template, and path. The label provides a description of the table. The template name displays the name of the template used to format the table. The path shows the output hierarchy to which the table belongs.

The fully qualified path is given in the trace record. A partially qualified path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, the full path for the parameter estimates for the first model in the preceding regression analysis is

```
Autoreg.Model1.OLSEst.ParameterEstimates
```

Therefore, partially qualified paths for the table are

```
Autoreg.Model1.OLSEst.ParameterEstimates
Model1.OLSEst.ParameterEstimates
OLSEst.ParameterEstimates
ParameterEstimates
```

To refer to a table (in order to select or exclude it from display, for example), specify either the table name or use the table's fully or partially qualified path. You may want to use qualified paths when your SAS program creates several tables that have the same name, as in the preceding example. In such a case, you can use a partially qualified path to select a subset of tables, or you can use a fully qualified path to select a particular table.

You specify the tables that ODS selects or excludes with the ODS SELECT or ODS EXCLUDE statement. Suppose that you want to display only the tables of parameter estimates from the preceding regression analysis. You can give any of the following statements (before invoking the AUTOREG procedure) to display both tables of parameter estimates. For this example, these statements are equivalent:

```
ods select Autoreg.Model1.OLSEst.ParameterEstimates
           Autoreg.Model2.OLSEst.ParameterEstimates;

ods select Model1.OLSEst.ParameterEstimates
           Model2.OLSEst.ParameterEstimates;

ods select OLSEst.ParameterEstimates;

ods select ParameterEstimates;
```

The first ODS SELECT statement specifies the full path for both tables. The second statement specifies the partially qualified path for both tables. The third and fourth statements specify the partial path "OLSEst.ParameterEstimates," and single name "ParameterEstimates," which are shared by both tables.

The Output Delivery System records the specified table names in its internal selection or exclusion list. ODS then processes the output it receives. Note that ODS maintains an overall selection or exclusion list that pertains to all ODS destinations, and it maintains a separate selection or exclusion list for each ODS destination. The list for a specific destination provides the primary filtering step. Restrictions you specify in the overall list are added to the destination-specific lists.

Suppose, for example, that your listing exclusion list (that is, the list of tables you wish to exclude from the SAS listing) contains the "Summary" table, which you specify with the statement

```
ods listing exclude Summary;
```

and your overall selection list (that is, the list of tables you want to select for all destinations) contains the tables "Summary" and "ParameterEstimates," which you specify with the statement

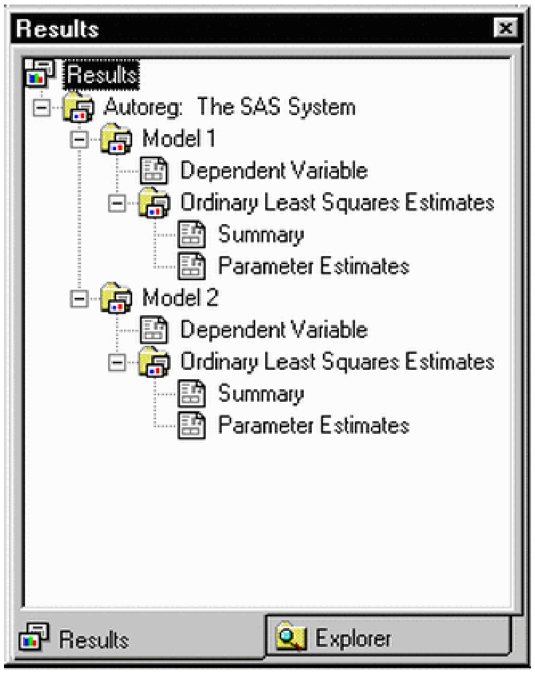
```
   ods select ParameterEstimates Summary;
```

The Output Delivery System then sends only the "ParameterEstimates" and "Summary" tables to all open destinations except the SAS listing. It sends only the "ParameterEstimates" table to the SAS listing because the table "Summary" is excluded from that destination.

Some SAS procedures, such as the ARIMA or the MODEL procedure, support run-group processing, which means that a RUN statement does not end the procedure. A QUIT statement explicitly ends such procedures; if you omit the QUIT statement, a PROC or a DATA statement implicitly ends such procedures. When you use the Output Delivery System with procedures that support run-group processing, it is good programming practice to specify a QUIT statement at the end of the procedure. This causes ODS to clear the selection or exclusion list, and you are less likely to encounter unexpected results.

Using ODS with the SAS Explorer

The SAS Explorer is a new feature that enables you to examine the various parts of the SAS System. Figure 8.2 displays the Results window from the SAS Explorer. The Results node retains a running record of your output as it is generated during your SAS session. Figure 8.2 displays the output hierarchy when the preceding statements are executed.



```
Results
Results
  Autoreg: The SAS System
    Model 1
      Dependent Variable
      Ordinary Least Squares Estimates
        Summary
        Parameter Estimates
    Model 2
      Dependent Variable
      Ordinary Least Squares Estimates
        Summary
        Parameter Estimates
```

Figure 8.2. The Results Window from the SAS Explorer

When you click on the output table names in the Results window, you link directly to the output in the output window or, if you specify the HTML destination, in an HTML browser. The items on the left-hand side of the Results node are output directories. The items on the right-hand side of the Results node are the names of the actual output objects. You can also use the Explorer to determine names of the templates associated with each output table.

Controlling Output Appearance with Templates

A template is an abstract description of how output should appear when it is formatted. Templates describe several characteristics of the output, including headers, column ordering, style information, justification, and formats. All SAS/ETS procedures have templates, which are stored in the SASHELP library.

You can create or modify a template with the TEMPLATE procedure. For example, you can specify different column headings or different orderings of columns in a table. You can find the template associated with a particular output table by using the ODS TRACE statement or the SAS Explorer.

You can display the contents of a template by executing the following statements:

```
proc template;
   source  templatename;
run;
```

where *templatename* is the name of the template.

Suppose you want to change the way all of the parameter estimates are displayed by the AUTOREG procedure. You can redefine the templates that the procedure uses with PROC TEMPLATE. For example, in order to have the ESTIMATE and STANDARD ERROR columns always displayed with more digits, you can redefine the columns used by the procedure to display them:

```
proc template;
   edit ets.autoreg.ParameterEstimates;
      edit Estimate; format=Best16.; end;
      edit StdErr; format=Best16.; end;
   end;
run;
```

The BEST*w*. format enables you to display the most information about a value, according to the available field width. The BEST16. format specifies a field width of 16. Refer to the chapter on formats in *SAS Language Reference: Dictionary* for detailed information.

When you run PROC TEMPLATE to modify or edit a template, the template is stored in your SASUSER library. You can then modify the path that ODS uses to look up templates with the ODS PATH statement in order to access these new templates in a later SAS session. This means that you can create a default set of templates to modify the presentation format for all your SAS output. (Note that you can specify the SHOW option in the ODS PATH statement to determine the current path.)

It is important to note the difference between a style template and a table template. A table template applies only to the specific tables that reference the template. The preceding statements that modify the "ets.autoreg.ParameterEstimates" template provide an example of modifying columns within a table template.

A style template applies to an entire SAS job and can be specified only in the ODS HTML statement. You can specify a style as follows:

```
ods html style=Styles.Brown;
```

A style template controls stylistic elements such as colors, fonts, and presentation attributes. When you use a style template, you ensure that all your output shares a consistent presentation style.

You can also reference style information in table templates for individual headers and data cells. You can modify either type of template with the TEMPLATE procedure. For information on creating your own styles, refer to *SAS Output Delivery System User's Guide*.

Interaction Between ODS and the NOPRINT Option

Most SAS/ETS procedures support a NOPRINT option that you can use when you want to create an output data set but do not want any displayed output. Typically, you use an OUTPUT statement in addition to the procedure's NOPRINT option to create a data set and suppress displayed output.

You can also use the Output Delivery System to create output data sets by using the ODS OUTPUT statement. However, if you specify the NOPRINT option, the procedure may not send any output to the Output Delivery System. Therefore, when you want to create output data sets through ODS (using the ODS OUTPUT statement), and you want to suppress the display of all output, specify

```
ODS SELECT NONE;
```

or close the active ODS destinations by giving the command

```
ODS  destinationname CLOSE;
```

where *destinationname* is the name of the active ODS destination (for example, ODS HTML CLOSE).

Note: The ODS statement does not instruct a procedure to generate output: instead, it specifies how the Output Delivery System should manage the table once it is created. The requested data table (output) has to be generated by the procedure before ODS can manage it. You must ensure that the proper options are in effect. For example, the following code does not create the requested data set Parms.

```
proc autoreg;
   ods output ML.ParameterEstimates=Parms;
   model y1 = time;
run;
```

When you execute these statements, the following line is displayed in the log:

```
WARNING: Output 'ML.ParameterEstimates' was not created.
```

The data set Parms is not created because the table of parameter estimates is generated only when the METHOD=ML option is specified in the MODEL statement in the AUTOREG procedure.

Compatibility Issues with Version 6 Prototypes

- The Version 6 prototype of the ODS output hierarchy is stored in a SAS catalog. The latest version of SAS software has a more flexible item-store file type used to store templates and ODS output.
- The Version 6 prototype ODS uses two macro variables (_DISK_ and _PRINT_) to regulate the saving of an output hierarchy. The latest version of SAS software uses the global ODS statement to accomplish this task.
- The Version 6 PROC TEMPLATE and PROC OUTPUT syntax is not compatible with the latest version of SAS software.

Examples

The following examples display typical uses of the Output Delivery System.

Example 8.1. Creating HTML Output with ODS

This example demonstrates how you can use the ODS HTML statement to display your output in hypertext markup language (HTML).

The following statements create the data set AR2, which contains a second-order autocorrelated time series Y. The AUTOREG procedure is then invoked to estimate the time trend of Y.

The ODS HTML statement specifies the name of the file to contain body of the HTML output.

```
data AR2;
   ul = 0; ull = 0;
   do Time = -10 to 36;
      u = + 1.3 * ul - .5 * ull + 2*rannor(12346);
      Y = 10 + .5 * time + u;
      if Time > 0 then output;
      ull = ul; ul = u;
   end;
run;

ods html body='trend.htm';

title 'Estimated Time Trend of Y';
proc autoreg;
   model Y = Time;
run;
ods html close;
```

By default, the SAS listing receives all output generated during your SAS run. In this example, the ODS HTML statement opens the HTML destination, and both destinations receive the generated output. Output 8.1.1 displays the results as they are displayed in the SAS listing.

Note that you must specify the following statement before you can view your output in a browser.

```
   ods html close;
```

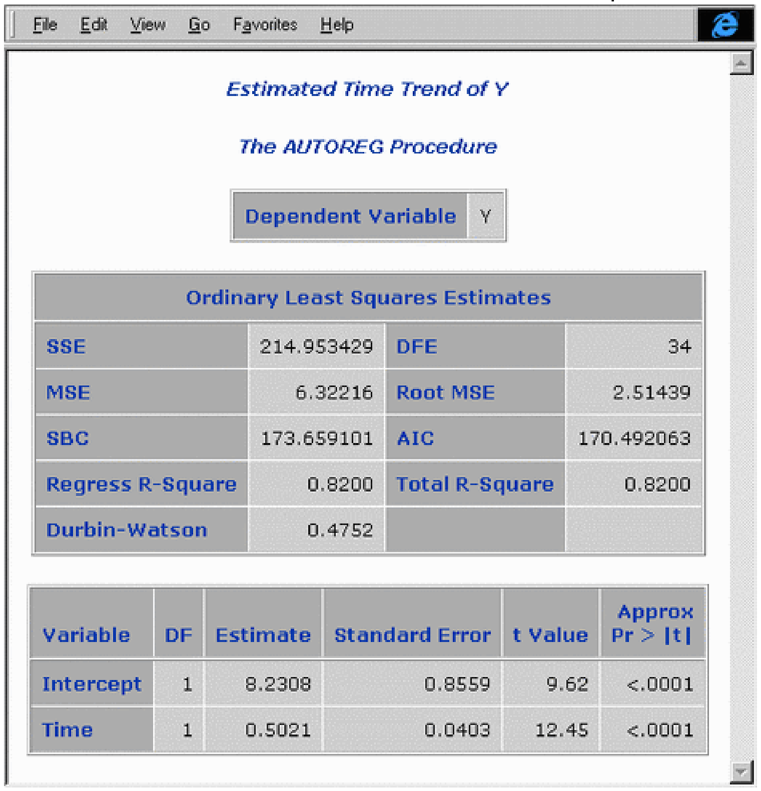
If you do not close the HTML destination, your HTML file may contain no output, or you may experience other unexpected results.

Output 8.1.2 displays the file 'trend.htm', which is specified in the preceding ODS HTML statement.

Output 8.1.1. Results for PROC AUTOREG: SAS Listing Output

```
                              Estimated Time Trend of Y

                                The AUTOREG Procedure

                              Dependent Variable    Y


                            Ordinary Least Squares Estimates

        SSE                   214.953429    DFE                       34
        MSE                      6.32216    Root MSE             2.51439
        SBC                   173.659101    AIC               170.492063
        Regress R-Square          0.8200    Total R-Square        0.8200
        Durbin-Watson             0.4752


                                             Standard                 Approx
     Variable        DF      Estimate          Error    t Value    Pr > |t|

     Intercept        1        8.2308         0.8559       9.62      <.0001
     Time             1        0.5021         0.0403      12.45      <.0001
```

Output 8.1.2. Results for PROC AUTOREG: HTML Output



Estimated Time Trend of Y

The AUTOREG Procedure

Dependent Variable	Y

Ordinary Least Squares Estimates			
SSE	214.953429	DFE	34
MSE	6.32216	Root MSE	2.51439
SBC	173.659101	AIC	170.492063
Regress R-Square	0.8200	Total R-Square	0.8200
Durbin-Watson	0.4752		

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.2308	0.8559	9.62	<.0001
Time	1	0.5021	0.0403	12.45	<.0001

Example 8.2. Creating HTML Output with a Table of Contents

The following example uses ODS to display the output in HTML with a table of contents.

The data are the population of the United States in millions recorded at ten year intervals starting in 1790 and ending in 1990. The MODEL procedure is used to estimate a logistic growth curve by nonlinear ordinary least squares.

```
data uspop;
   input pop :6.3 @@;
   retain year 1780;
   year=year+10;
   label pop='U.S. Population in Millions';
   datalines;
3929   5308   7239   9638   12866  17069  23191  31443
39818  50155  62947  75994  91972  105710 122775 131669
151325 179323 203211 226542 248710
;

ods html body='uspop.htm'
         contents='uspopc.htm'
         frame='uspopf.htm';

title 'Logistic Growth Curve Model of U.S. Population';
```

```
proc model data=uspop;
   label a = 'Maximum Population'
         b = 'Location Parameter'
         c = 'Initial Growth Rate';
   pop = a / ( 1 + exp( b - c * (year-1790) ) );
   fit pop start=(a 1000  b 5.5  c .02)/ out=resid outresid;
run;
ods html close;
```

The ODS HTML statement specifies three files. The BODY= option specifies the file to contain the output generated from the statements that follow. The BODY= option is the only required option.

The CONTENTS= option specifies a file to contain the table of contents. The FRAME= option specifies a file to contain both the table of contents and the output. You open the FRAME= file in your browser to view the table of contents together with the generated output (see Output 8.2.1). Note that, if you specify the ODS HTML statement with only the BODY= argument, no table of contents is created.

The MODEL procedure is invoked to fit the specified model. The resulting output is displayed in Output 8.2.1.

Output 8.2.1. HTML Output from the MODEL Procedure

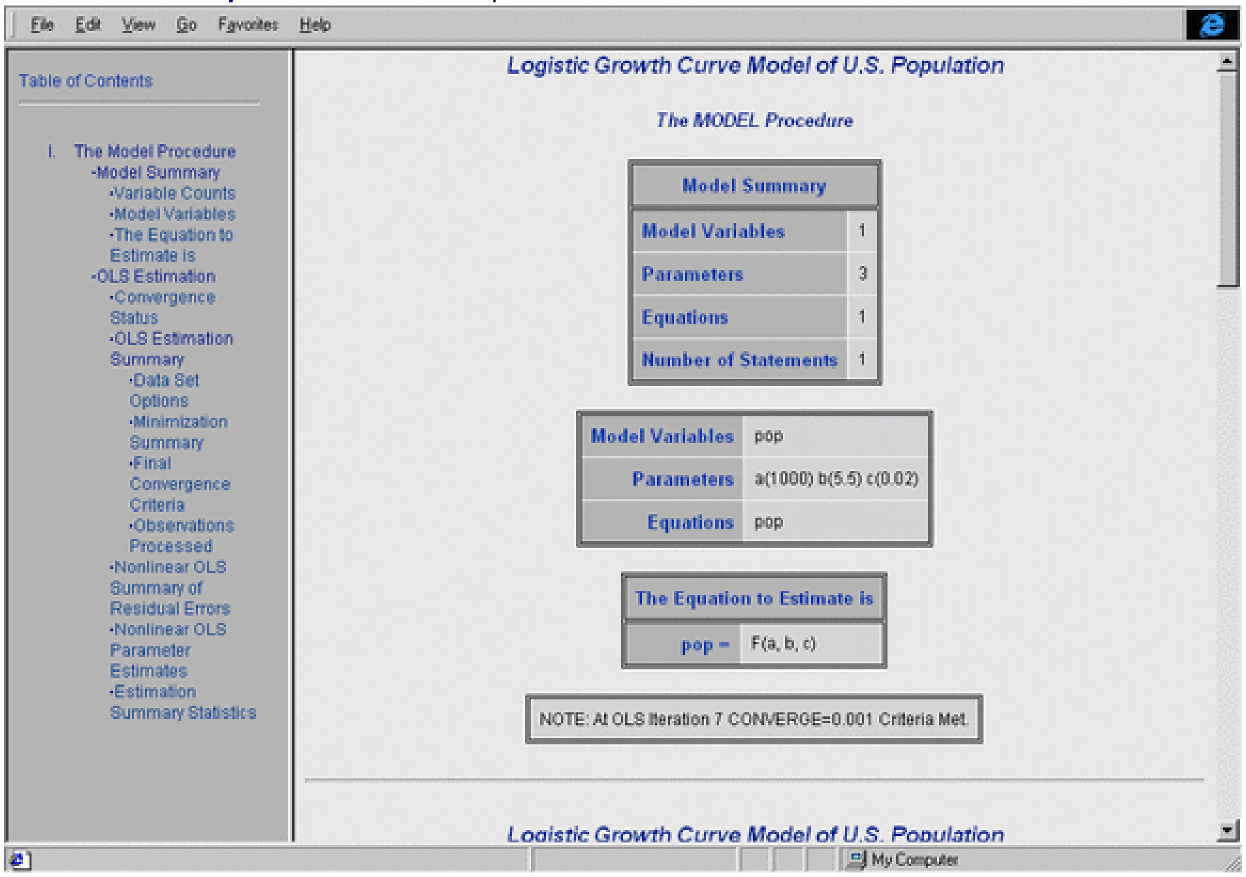


Table of Contents

I. The Model Procedure
- Model Summary
  - Variable Counts
  - Model Variables
  - The Equation to Estimate is
- OLS Estimation
  - Convergence Status
  - OLS Estimation Summary
    - Data Set Options
    - Minimization Summary
    - Final Convergence Criteria
    - Observations Processed
  - Nonlinear OLS Summary of Residual Errors
  - Nonlinear OLS Parameter Estimates
  - Estimation Summary Statistics

Logistic Growth Curve Model of U.S. Population

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	3
Equations	1
Number of Statements	1

Model Variables	pop
Parameters	a(1000) b(5.5) c(0.02)
Equations	pop

The Equation to Estimate is	
pop =	F(a, b, c)

NOTE: At OLS Iteration 7 CONVERGE=0.001 Criteria Met.

Logistic Growth Curve Model of U.S. Population

The table of contents displayed in Output 8.2.1 contains the descriptive label for each output table produced in the MODEL procedure step. You can select any label in the table of contents and the corresponding output will be displayed in the right-hand side of the browser window.

Example 8.3. Determining the Names of ODS Tables

In order to select or exclude a table, or to render it as a SAS data set, you must first know its name. You can obtain the table names in several ways:

- For any SAS/ETS procedure, you can obtain table names from the individual procedure chapter or from the SAS online Help system.
- For any SAS procedure, you can use the SAS Explorer window to view the names of the tables created in your SAS run.
- For any SAS procedure, you can use the ODS TRACE statement to find the names of tables created in your SAS run. The ODS TRACE statement writes identifying information to the SAS log for each generated output table.

This example uses the ODS TRACE statement with the LISTING option to obtain the names of the created output objects. By default, the ODS TRACE statement writes its information to the SAS log. However, you can specify the LISTING option to have the information interleaved with the procedure output in the SAS listing.

The model will be the U.S. population model from the previous example.

```
ods trace on/listing;

title 'Logistic Growth Curve Model of U.S. Population';
proc model data=uspop;
   label a = 'Maximum Population'
         b = 'Location Parameter'
         c = 'Initial Growth Rate';
   pop = a / ( 1 + exp( b - c * (year-1790) ) );
   fit pop start=(a 1000  b 5.5  c .02)/ out=resid outresid;
run;

ods trace off;
```

The purpose of these statements is to obtain the names of the ODS tables produced in this PROC MODEL run. The ODS TRACE ON statement writes the trace record of ODS output tables. The LISTING option specifies that the information is interleaved with the output and written to the SAS listing.

The MODEL procedure is invoked to perform the analysis, the SAS listing receives the procedure output and the trace record, and the trace is then turned off with the OFF option.

Output 8.3.1. The ODS Trace, Interleaved with MODEL Results: Partial Results

```
                                   The MODEL Procedure

Output Added:
-------------
Name:       ResidSummary
Label:      Nonlinear OLS Summary of Residual Errors
Template:   ets.model.ResidSummary
Path:       Model.OLS.ResidSummary
-------------


                              Nonlinear OLS Summary of Residual Errors

                 DF     DF                                              Adj
Equation      Model  Error        SSE        MSE  Root MSE  R-Square     R-Sq  Label

pop               3     18      345.6    19.2020    4.3820    0.9972   0.9969  U.S. Population
                                                                                in Millions

Output Added:
-------------
Name:       ParameterEstimates
Label:      Nonlinear OLS Parameter Estimates
Template:   ets.model.ParameterEstimates
Path:       Model.OLS.ParameterEstimates
-------------


                                 Nonlinear OLS Parameter Estimates

                                           Approx                     Approx
      Parameter       Estimate      Std Err    t Value     Pr > |t|    Label

      a               387.9307      30.0404      12.91       <.0001    Maximum Population
      b               3.990385       0.0695      57.44       <.0001    Location Parameter
      c               0.022703      0.00107      21.22       <.0001    Initial Growth Rate
```

As displayed in Output 8.3.1, the ODS TRACE ON statement writes the name, label, template, and path name of each generated ODS table. For more information on names, labels, and qualified path names, see the discussion in the section "Using the Output Delivery System" beginning on page 244.

The information obtained with the ODS TRACE ON statement enables you to request output tables by name. The examples that follow demonstrate how you can use this information to select, exclude, or create data sets from particular output tables.

Example 8.4. Selecting ODS Tables for Display

You can use the ODS SELECT statement to deliver only certain tables to open ODS destinations. In the following example, the MODEL procedure is used to fit a model for new one-family home sales.

```
title 'Modeling One-Family Home Sales';
    data homes;
       input year q pop yn cpi @@;
          y=yn/cpi;
       label q='New One-Family Houses Sold in Thousands'
          pop='U.S. Population in Millions'
          y='Real Personal Income in Billions'
          cpi='U.S. CPI 1982-1984 = 100';
```

```
   datalines;
70 485 205.052  715.6  .388  71 656 207.661  776.8  .405
72 718 209.896  839.6  .418  73 634 211.909  949.8  .444
74 519 213.854 1038.4  .493  75 549 215.973 1142.8  .538
76 646 218.035 1252.6  .569  77 819 220.239 1379.3  .606
78 817 222.585 1551.2  .652  79 709 225.055 1729.3  .726
80 545 227.719 1918.0  .824  81 436 229.945 2127.6  .909
82 412 232.171 2261.4  .965  83 623 234.296 2428.1  .996
84 639 236.343 2668.6 1.039  85 688 238.466 2838.7 1.076
86 750 240.658 3013.3 1.096  87 671 242.820 3194.7 1.136
88 676 245.051 3479.2 1.183  89 650 247.350 3725.5 1.240
90 536 249.975 3945.8 1.307
;

ods select ResidSummary ParameterEstimates;
ods trace on;
ods show;
```

The ODS SELECT statement specifies that only the two tables “ResidSummary” and “ParameterEstimates” are to be delivered to the ODS destinations. In this example, no ODS destinations are explicitly opened. Therefore, only the SAS listing, which is open by default, receives the procedure output. The ODS SHOW statement displays the current overall selection list in the SAS log. The ODS TRACE statement writes the trace record of the ODS output objects to the SAS log. In the following statements, the MODEL procedure is invoked to produce the output.

```
proc model data=homes;
   parms a b c d;
      q = a + b*y + c*lag(y) + d*pop;
   %ar(ar_q,1,q)
   endo q;
   exo y pop;
   id year;
   fit q / dw;
run;
```

Output 8.4.1 displays the results of the ODS SHOW statement, which writes the current overall selection list to the SAS log. As specified in the preceding ODS SELECT statement, only the two ODS tables “ResidSummary” and “ParameterEstimates” are selected for output.

Output 8.4.1. Results of the ODS SHOW Statement

```
     ods select ResidSummary ParameterEstimates;
     ods trace on;
     ods show;

Current OVERALL select list is:
1. ResidSummary
2. ParameterEstimates
```

Partial results of the ODS TRACE statement, which is written to the SAS log, are displayed in Output 8.4.2.

Output 8.4.2. The ODS TRACE: Partial Contents of the SAS Log

```
      proc model data=homes;
         parms a b c d;
            q = a + b*y + c*lag(y) + d*pop;
         %ar(ar_q,1,q)
         endo q;
         exo y pop;
         id year;

         fit q / dw;
      run;


Output Added:
-------------
Name:       ResidSummary
Label:      Nonlinear OLS Summary of Residual Errors
Template:   ets.model.ResidSummary
Path:       Model.OLS.ResidSummary
-------------

Output Added:
-------------
Name:       ParameterEstimates
Label:      Nonlinear OLS Parameter Estimates
Template:   ets.model.ParameterEstimates
Path:       Model.OLS.ParameterEstimates
-------------
```

In the following statements, the ODS SHOW statement writes the current overall selection list to the SAS log. The QUIT statement ends the MODEL procedure. The second ODS SHOW statement writes the selection list to the log after PROC MODEL terminates. The ODS selection list is reset to ’ALL,’ by default, when a procedure terminates. For more information on ODS exclusion and selection lists, see the section “Using the Output Delivery System” beginning on page 244.

```
   ods show;
   quit;
   ods show;
```

The results of the statements are displayed in Output 8.4.3. Before the MODEL procedure terminates, the ODS selection list includes only the two tables, “ResidSummary” and “ParameterEstimates.”

Output 8.4.3. The ODS Selection List, Before and After PROC MODEL Terminates

```
     ods show;

Current OVERALL select list is:
1. ResidSummary
2. ParameterEstimates


     quit;

NOTE: PROCEDURE MODEL used:
      real time           0.34 seconds
      cpu time            0.19 seconds


     ods show;

Current OVERALL select list is: ALL
```

The MODEL procedure supports run-group processing. Before the QUIT statement is executed, PROC MODEL is active and the ODS selection list remains at its previous setting before PROC MODEL was invoked. After the QUIT statement, the selection list is reset to deliver all output tables.

The entire displayed output consists of the two selected tables, as displayed in Output 8.4.4.

Output 8.4.4. The Listing Output of the ResidSummary and ParameterEstimates Tables from PROC MODEL

```
                       Logistic Growth Curve Model of U.S. Population

                                   The MODEL Procedure

                         Nonlinear OLS Summary of Residual Errors

                  DF       DF                                                 Adj      Durbin
 Equation       Model    Error         SSE          MSE     R-Square        R-Sq      Watson

 q                  5       15     86388.2       5759.2       0.6201      0.5188      1.7410


                            Nonlinear OLS Parameter Estimates

                                   Approx                  Approx
     Parameter       Estimate     Std Err    t Value     Pr > |t|    Label

     a               2622.538      1196.5       2.19       0.0446
     b               1.216858      0.3723       3.27       0.0052
     c               -0.65809      0.3676      -1.79       0.0936
     d               -14.8418      8.6435      -1.72       0.1065
     ar_q_l1         0.478075      0.2480       1.93       0.0730    AR(ar_q) q lag1
                                                                     parameter
```

Example 8.5. Creating an Output Data Set from an ODS Table

The ODS OUTPUT statement creates SAS data sets from ODS tables. In the following example, the AUTOREG procedure is invoked to estimate a large number of Dickey-Fuller type regressions and part of the resulting procedure output is output to a SAS data set. The Dickey-Fuller t-statistic is then calculated and PROC MEANS is used to calculate the empirical critical values.

The data set UNITROOT contains 10,000 unit root time series.

```
data unitroot;
  YLag = 0;
  do rep = 1 to 10000;
    do time = -50 to 100;
      Y = YLag + rannor(123);
      if time > 0 then output;
      YLag = Y;
    end;
  end;
run;
```

Determining the Names of the ODS Tables

The purpose of the following statements is to obtain the names of the output tables produced in this PROC AUTOREG run. Note that a smaller data set, test, is used for this trial run. The ODS TRACE statement lists the trace record.

```
data test;
  YLag = 0;
    do time = -50 to 100;
      Y = YLag + rannor(123);
      if time > 0 then output;
      YLag = Y;
    end;
run;

ods trace on;
proc autoreg data=test;
   model Y = YLag;
run;
ods trace off;
```

Output 8.5.1. The ODS TRACE: Partial Contents of the SAS Log

```
     ods trace on;
     ods listing close;
     proc autoreg data=test;
        model Y = YLag;
     run;


Output Added:
-------------
Name:       Dependent
Label:      Dependent Variable
Template:   ets.autoreg.Dependent
Path:       Autoreg.Model1.Dependent
-------------

.
.
.

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   ets.autoreg.ParameterEstimates
Path:       Autoreg.Model1.OLSEst.ParameterEstimates
-------------
```

By default, the trace record is written to the SAS log, as displayed in Output 8.5.1. Note that you can alternatively specify that the information be interleaved with the procedure output in the SAS listing (see Example 8.3).

Creating the Output Data Set

In the statements that follow, the ODS OUTPUT statement writes the ODS table "ParameterEstimates" to a SAS data set called myParms. All of the usual data set options, such as the KEEP= or WHERE= options, can be used in the ODS OUTPUT statement. Thus, to modify the ParameterEstimates data set so that it contains only certain variables, you can use the data set options as follows.

```
ods listing close;
proc autoreg data=unitRoot;
   ods output ParameterEstimates = myParms
           (keep=Variable Estimate StdErr
            where=(Variable='YLag')) ;
   by rep;
   model Y = YLag;
run;
ods listing;
```

The KEEP= option in the ODS OUTPUT statement specifies that only the variables Variable, Estimate, and StdErr are written to the data set. The WHERE= option selects the specific variable in which we are interested , YLag. The AUTOREG procedure is again invoked. In order to limit the amount of displayed output, the ODS exclusion list is set to ALL.

In the following statements, the output data set myParms is used to create the data set TDISTN which contains the Dickey-Fuller t-statistics. PROC MEANS is then utilized to tabulate the empirical 1, 5, and 10 percent critical values. The results are displayed in Output 8.5.2.

```
data tdistn;
  set myParms;
  tStat = (Estimate-1)/StdErr;
run;

ods select Means.Summary;
proc means data=tDistn P1 P5 P10 fw=5;
   var tStat;
   title 'Simulated Dickey-Fuller Critical Values';
run;
```

Output 8.5.2. The Empirical Critical Values, Tabulated by PROC MEANS

```
                    Simulated Dickey-Fuller Critical Values

                             The MEANS Procedure

                          Analysis Variable : tStat

                             1st      5th     10th
                            Pctl     Ptcl     Pctl
                           -----------------------
                           -3.51    -2.90    -2.59
                           -----------------------
```

Chapter 9
Statistical Graphics Using ODS (Experimental)

Chapter Contents

Chapter 9
Statistical Graphics Using ODS (Experimental)

Overview

Graphics are indispensable for modern statistical analysis. They enrich the analysis by revealing patterns, identifying differences, and expressing uncertainty that would not be readily apparent in tabular output. Effective graphics also add visual clarity to an analytical presentation, and they provoke questions that would not otherwise be raised, stimulating deeper investigation.

In SAS 9.1, a number of SAS/ETS procedures have been modified to use an experimental extension to the Output Delivery System (ODS) that enables them to create statistical graphics as automatically as tables. This facility is referred to as *ODS Statistical Graphics* (or *ODS Graphics* for short), and it is invoked when you provide the experimental ODS GRAPHICS statement prior to your procedure statements. Any procedures that use ODS Graphics then create graphics, either by default or when you specify procedure options for requesting specific graphs.

With ODS Graphics, a procedure creates the graphs that are most commonly needed for a particular analysis. In many cases, graphs are automatically enhanced with useful statistical information or metadata, such as sample sizes and p-values, which are displayed in an inset box. Using ODS Graphics eliminates the need to save numerical results in an output data set, manipulate them with a DATA step program, and display them with a graphics procedure.

The SAS/ETS procedures that use ODS Graphics in SAS 9.1 are listed on page 297. The plots produced by each procedure and any corresponding options are described in the procedure chapter. See the “ODS Graphics” subsection in the “Details” section of each procedure chapter for additional information.

In many ways, creating graphics with ODS is analogous to creating tables with ODS. You use

- procedure options and defaults to determine which graphs are created
- ODS destination statements (such as ODS HTML) to specify the output destination for graphics

Additionally, you can use

- graph names in ODS SELECT and ODS EXCLUDE statements to select or exclude graphs from your output
- ODS styles to control the general appearance and consistency of *all graphs*
- ODS templates to control the layout and details of *individual graphs*. A default template is provided by SAS for each graph.

In SAS 9.1, the ODS destinations that support ODS Graphics include HTML, LATEX, PRINTER, and RTF. These are discussed on page 274.

Both tables and graphs are saved in the ODS output file produced for a destination. However, individual graphs can also be saved in files, which are produced in a specific graphics image file type, such as GIF or PostScript. This enables you to access individual graphs for inclusion in a document. For example, you can save graphs in PostScript files to include in a paper that you are writing with LaTeX. Likewise, you can save graphs in GIF files to include in an HTML document. With the HTML destination, you can also request an image map format that supports tool tip displays, which appear when you move a mouse over certain features of the graph.

In common applications of procedures that use ODS Graphics, the default graphs should suffice. However, when modifications become necessary, you can customize a particular graph by changing its template, or you can make consistent changes to all your graphs by selecting a different ODS style or by modifying an existing ODS style definition:

- As with table definitions, you can access graph template definitions and modify them with the TEMPLATE procedure. Graph template definitions are written in an experimental graph template language, which has been added to the TEMPLATE procedure in SAS 9.1. This language includes statements for specifying plot types (such as scatter plots and histograms), plot layouts, and text elements (such as titles and insets). It also provides support for built-in computations (such as histogram binning) and evaluation of computational expressions. Options are available for specifying colors, marker symbols, and other aspects of plot features.
- ODS style definitions include a number of graph elements that correspond to general features of statistical graphics, such as titles and fitted lines. The attributes of these elements, such as fonts and colors, provide the defaults for options in graph templates provided by SAS. Consequently, you can change all of your graphs in a consistent manner by simply selecting a different style. For example, by specifying the “Journal” style, you can create gray-scale graphs and tables that are suitable for publication in professional journals.

Note: Statistical graphics created with ODS are experimental in this release, meaning that both their appearance and their syntax are subject to change in a future release.

This chapter illustrates the use of ODS Graphics, and it provides general information on managing your graphics. If you are unfamiliar with ODS, you will find it helpful to read Chapter 8, “Using the Output Delivery System.” For complete documentation on the Output Delivery System, refer to the *SAS Output Delivery System User’s Guide*.

How to Use This Chapter

If you are trying out ODS Graphics for the first time, begin by reading the section "Getting Started" on page 267, which provides the essentials. Additional examples are given in the chapters for procedures that use ODS Graphics in SAS 9.1.

To take full advantage of ODS Graphics, you will need to learn more about ODS destinations, output files, and image file types for graphics, as well as ways to access and include individual graphs in reports and presentations. This is explained in the section "Managing Your Graphics" on page 274, the section "Graphics Image Files" on page 283, and the section "Examples" beginning on page 300.

If you need to customize a graph by modifying its template, read the section "Customizing Graphics with Templates" on page 287 and the series of examples beginning on page 312.

If you need to customize a style definition read the section "Styles for Graphics" on page 293 and the series of examples beginning on page 322.

Getting Started

This section introduces the use of ODS Graphics with two simple examples, which illustrate how the ODS GRAPHICS statement and an ODS destination statement are required to produce graphics. In the first example, no procedure options are required; basic graphics are produced by default. In the second example, procedure options are used to request specific plots.

Using the ODS GRAPHICS Statement

This example is taken from the "Getting Started" section of Chapter 20, "The MODEL Procedure." It illustrates a situation in which only the ODS GRAPHICS statement and a supported ODS destination are needed to create graphical displays.

The SASHELP library contains the data set CITIMON, which, in turn, includes the variable LHUR, the monthly unemployment figures, and the variable IP, the monthly industrial production index. Assume that these variables are related by the following nonlinear equation:

$$lhur = \frac{1}{a \cdot \mathrm{ip} + b} + c + \epsilon$$

In this equation *a*, *b*, and *c* are unknown coefficients and ϵ is an unobserved random error.

The following statements illustrate how to use PROC MODEL to estimate values for *a*, *b*, and *c* from the data in SASHELP.CITIMON.

```
ods html;
ods graphics on;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;

ods graphics off;
ods html close;
```

The ODS HTML statement specifies an HTML destination for the output. Note that the LISTING destination is not supported by ODS Graphics in SAS 9.1. For a discussion of ODS destinations that are supported, see page 274.

The ODS GRAPHICS statement is specified to request ODS Graphics in addition to the usual tabular output. Here, the graphical output consists of a studentized residual plot, a Cook's D plot, a plot of actual and predicted values, plots of the sample autocorrelation, partial autocorrelation, and inverse autocorrelation function of residuals, a QQ plot and a histogram of residuals; these are shown in Figure 9.1 through Figure 9.8, respectively.

The ODS GRAPHICS OFF statement disables ODS Graphics, and the ODS HTML CLOSE statement closes the HTML destination.

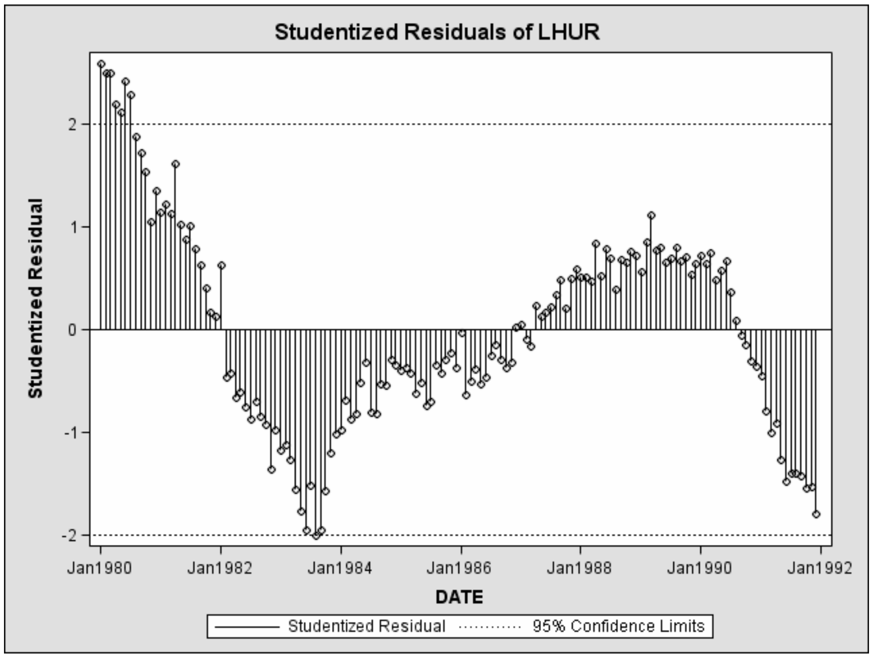


Figure 9.1. Studentized Residuals

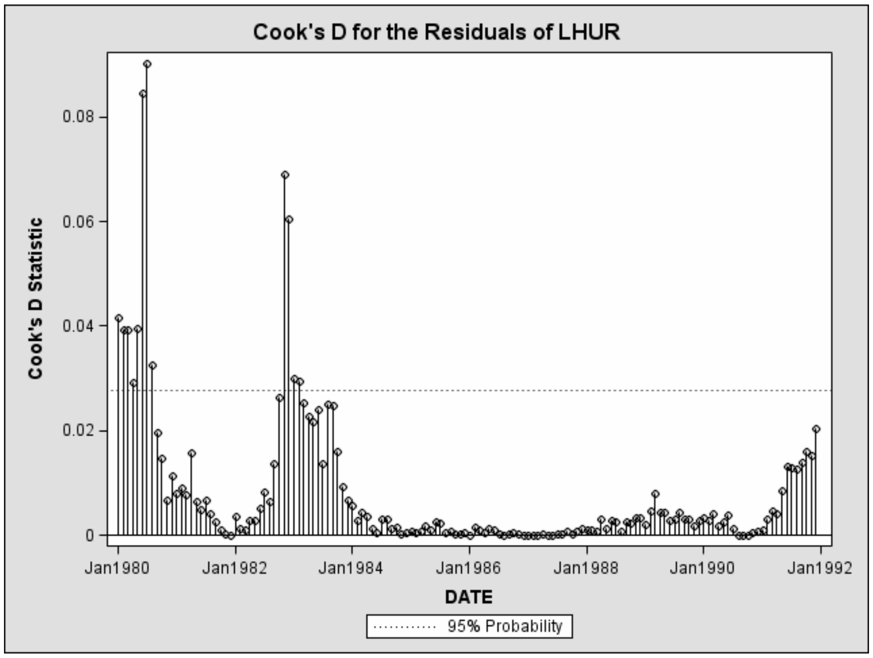


Figure 9.2. Cook's D for Residuals

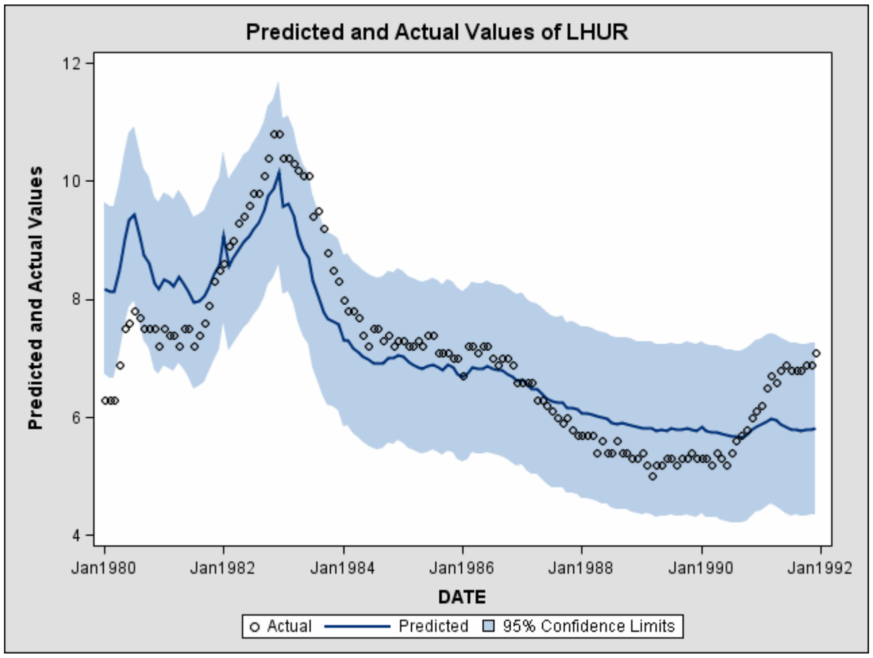


Figure 9.3. Predicted and Actual Values

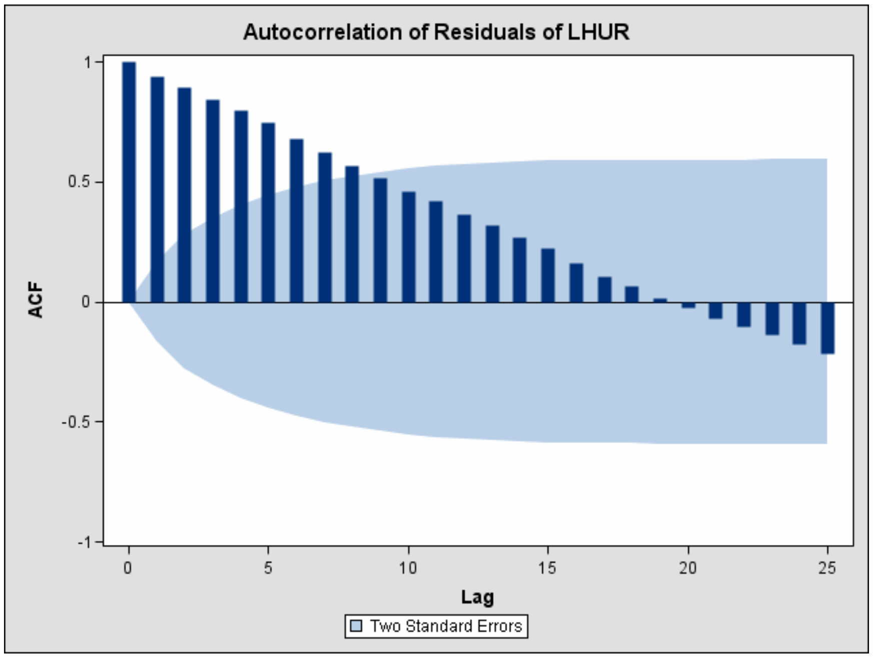


Figure 9.4. Autocorrelation of Residuals

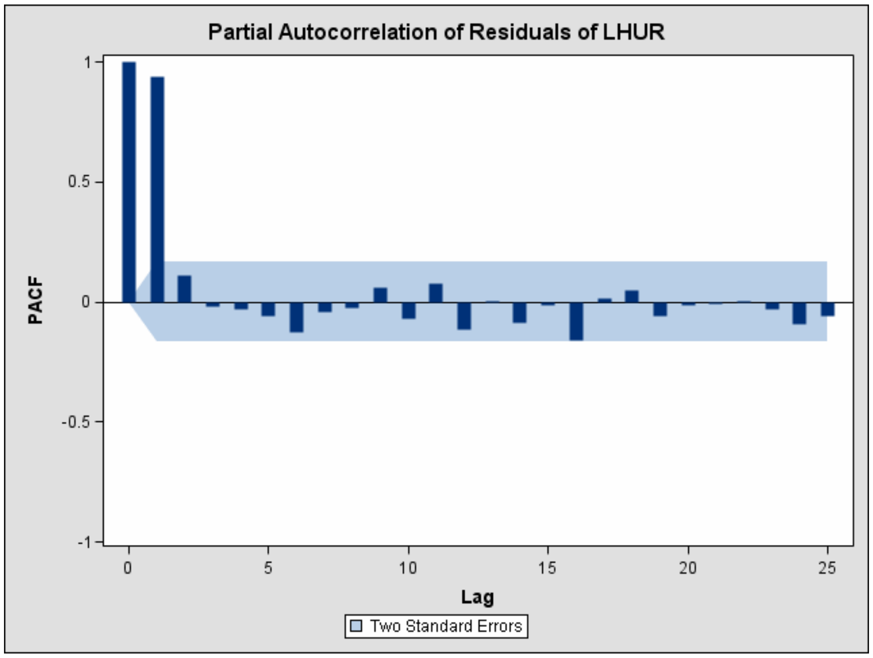


Figure 9.5. Partial Autocorrelation of Residuals

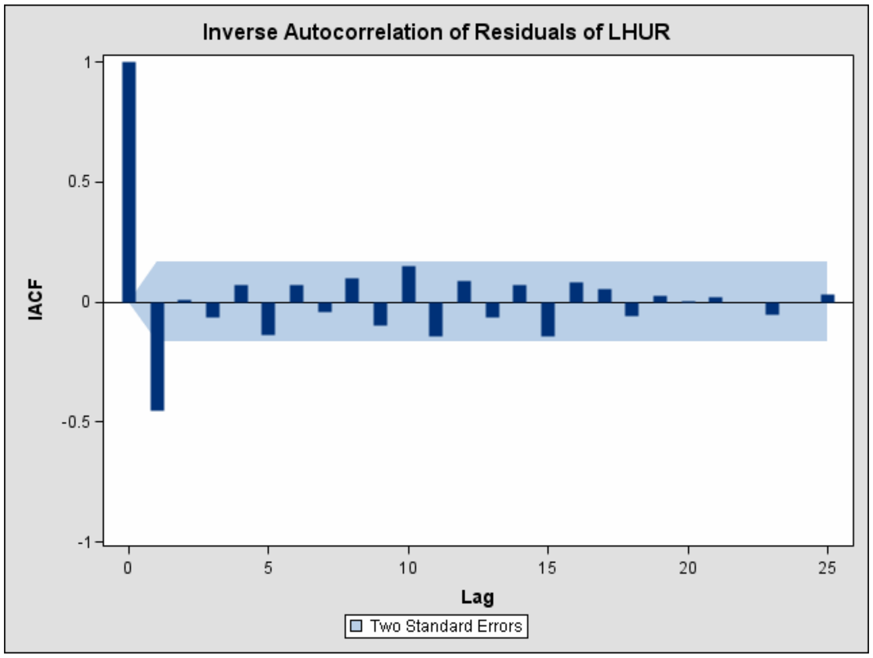


Figure 9.6. Inverse Autocorrelation of Residuals

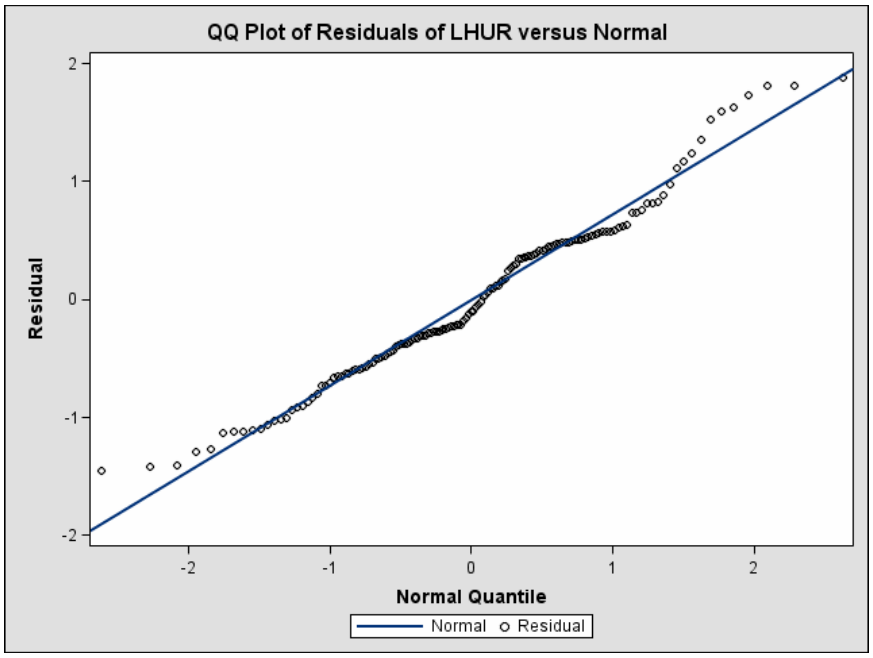


Figure 9.7. QQ Plot of Residuals

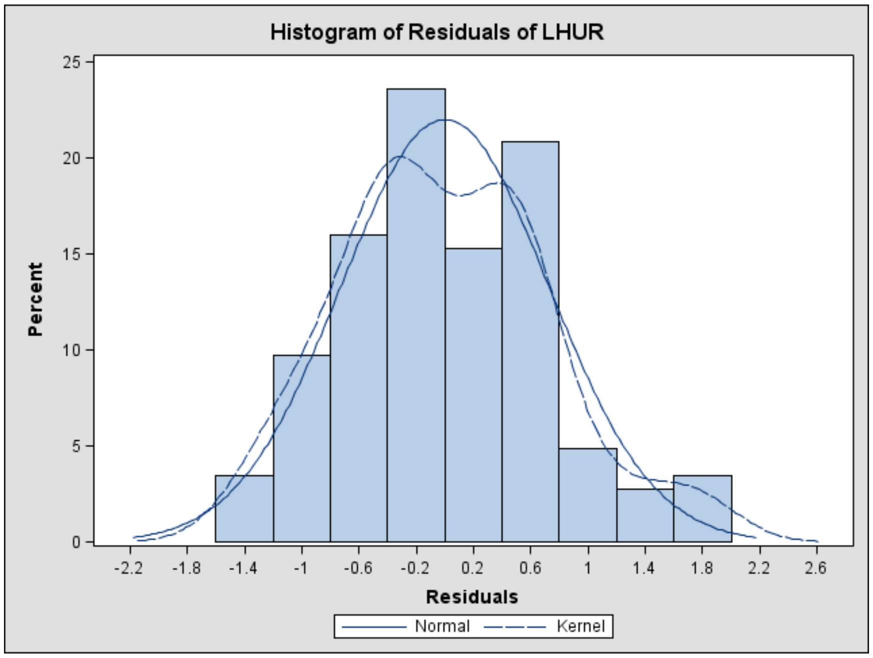


Figure 9.8. Histogram of Residuals

For more information about ODS Graphics available in the MODEL procedure, see the "ODS Graphics" section on page 1166 in Chapter 20, "The MODEL Procedure."

A sample program named odsgrgs.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Using the ODS GRAPHICS Statement and Procedure Options

In this example, new options of the UCM procedure are used to request graphical displays in addition to the ODS GRAPHICS statement.

The following data from the Connecticut Tumor Registry presents age-adjusted numbers of melanoma incidences per 100,000 people for 37 years from 1936 to 1972 The data have been used in Houghton, Flannery, and Viola (1980).

```
data melanoma;
   input Melanoma_Incidence_Rate @@;
   year = mdy(1,1, 1836 + _n_-1 );  /* start year 1936 */
   format year year4.;
   datalines;
   0.9 0.8 0.8 1.3 1.4 1.2 1.7 1.8 1.6 1.5
   1.5 2.0 2.5 2.7 2.9 2.5 3.1 2.4 2.2 2.9
   2.5 2.6 3.2 3.8 4.2 3.9 3.7 3.3 3.7 3.9
   4.1 3.8 4.7 4.4 4.8 4.8 4.8
   ;
run;
```

The following statements request the estimation and forecast of the following model and plots of the smoothed cycle component and forecasts.

```
Melanoma_Incidence_Rate = trend + cycle + error
```

```
ods html;
ods graphics on;

proc ucm data=melanoma noprint;
   id year interval=year;
   model Melanoma_Incidence_Rate;
   irregular;
   level variance=0 noest;
   slope variance=0 noest;
   cycle rho=1 noest=rho plot=smooth;
   estimate back=5;
   forecast back=5 lead=10 plot=forecasts print=none;
run;

ods graphics off;
ods html close;
```

The smoothed cycle component and forecasts plots are displayed in Figure 9.9 and Figure 9.10, respectively. This graphical display are requested by specifying the ODS GRAPHICS statement prior to the procedure statements, and the experimental PLOTS= options in the CYCLE and FORECAST statements. For more information about the graphics available in the UCM procedure, see the "ODS Graphics" section on page 1664 in Chapter 29, "The UCM Procedure."

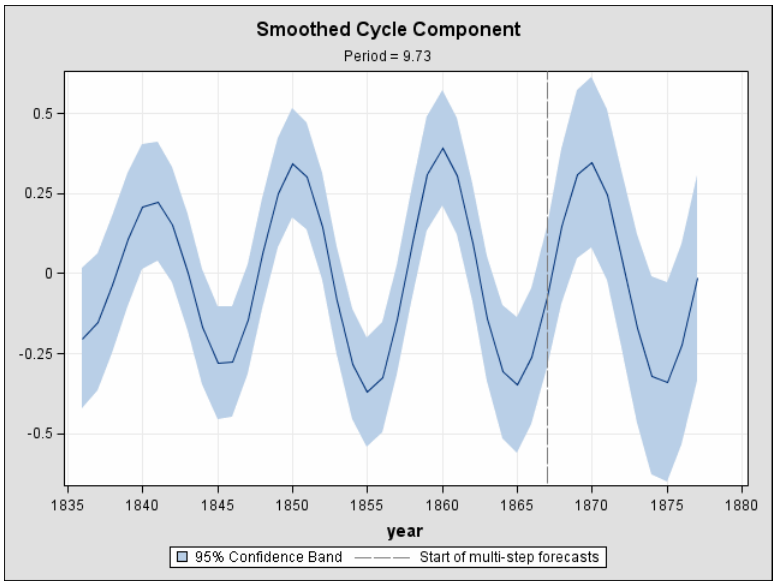


Figure 9.9. Smoothed Cycle Component Plot

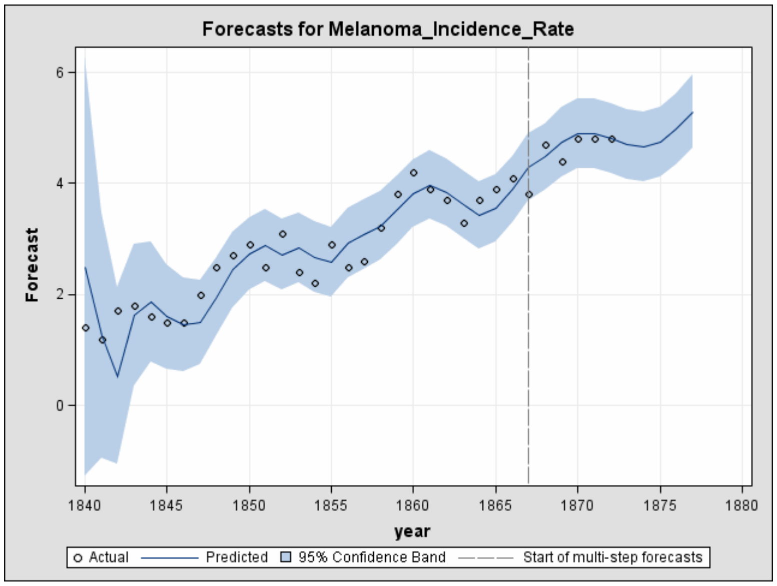


Figure 9.10. Forecasts Plot

A sample program named odsgrgs.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Managing Your Graphics

This section describes techniques for managing your graphics:

- specifying an ODS destination for graphics
- viewing your graphs in the SAS windowing environment
- referring to graphs by name when using ODS
- selecting and excluding graphs from your output
- modifying the appearance of all your graphs with styles

Specifying an ODS Destination for Graphics

Whenever you use ODS Graphics you must specify a valid ODS destination. The examples in "Getting Started" illustrate how to specify an HTML destination. Other destinations are specified in a similar way. For example, you can specify an RTF destination with the following statements.

```
ods rtf;
ods graphics on;

   ...SAS statements...

ods graphics off;
ods rtf close;
```

The supported ODS destinations are shown in Table 9.1.

Table 9.1. Destinations Supported by ODS Graphics

Destination	**Destination Family**	**Viewer**
DOCUMENT		Not Applicable
HTML	MARKUP	Browser
LATEX	MARKUP	Ghostview
PCL	PRINTER	Ghostview
PDF	PRINTER	Acrobat
PS	PRINTER	Ghostview
RTF		Microsoft Word

Note: In SAS 9.1 the LISTING destination does not support ODS Graphics. You must specify a supported ODS destination in order to produce ODS Graphics, as illustrated by all the examples in this chapter.

Specifying a File for ODS Output

You can specify a file name for your output with the FILE= option in the ODS destination statement, as in the following example:

```
ods html file = "test.htm";
```

The output is written to the file test.htm, which is saved in the SAS current folder. At startup, the SAS current folder is the same directory in which you start your SAS session. If you are running SAS with the windowing environment in the Windows operating system, then the current folder is displayed in the status line at the bottom of the main SAS window, as shown in Figure 9.11.

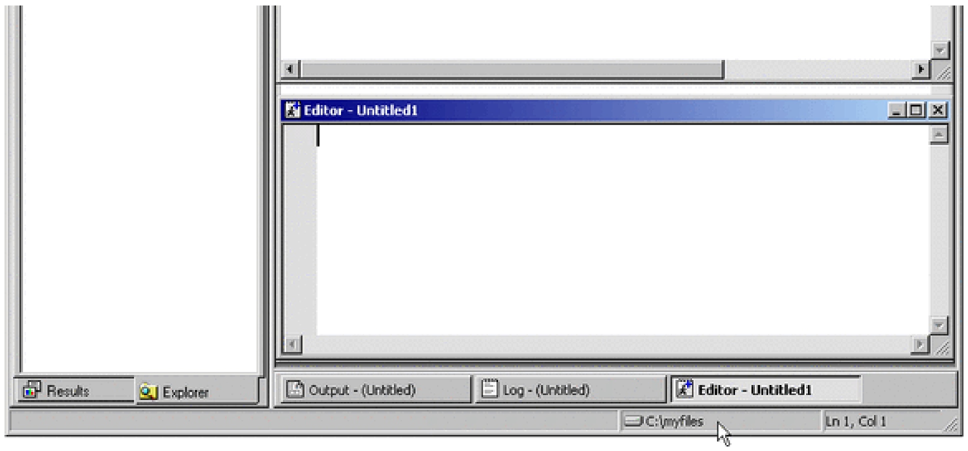


Figure 9.11. Current Folder (Right Bottom)

If you do not specify a file name for your output, then SAS provides a default file, which depends on the ODS destination. This file is saved in the SAS current folder. You can always check the SAS log to verify the name of the file in which your output is saved. For example, suppose you specify the following statement at startup:

```
ods html;
```

Then the following message is displayed in the SAS log:

```
NOTE: Writing HTML Body file: sashtml.htm
```

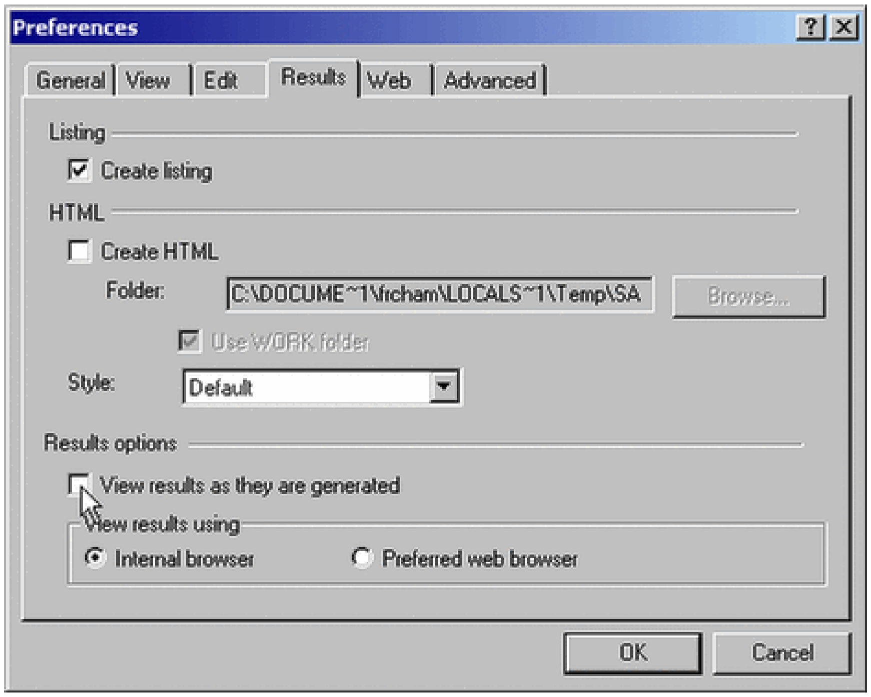
The default file names for each destination are specified in the SAS Registry. For more information, refer to the SAS Companion for your operating system.

Viewing Your Graphs in the SAS Windowing Environment

The mechanism for viewing graphics created with ODS can vary depending on your operating system, which viewers are installed on your computer, and the ODS destination you have selected.

If you are using the SAS windowing environment in the Windows operating system and you specify an HTML destination, then by default the results are displayed in the SAS Results Viewer as they are being generated. Depending on your configuration, this may also apply to the PDF and RTF destinations.* For information about the windowing environment in a different operating system, refer to the SAS Companion for that operating system.

If you do not want to view the results as they are being generated, then select **Tools** → **Options** → **Preferences...** from the menu at the top of the main SAS window. Then in the **Results** tab disable **View results as they are generated**, as shown in Figure 9.12.



Preferences

General | View | Edit | Results | Web | Advanced

Listing
[✓] Create listing

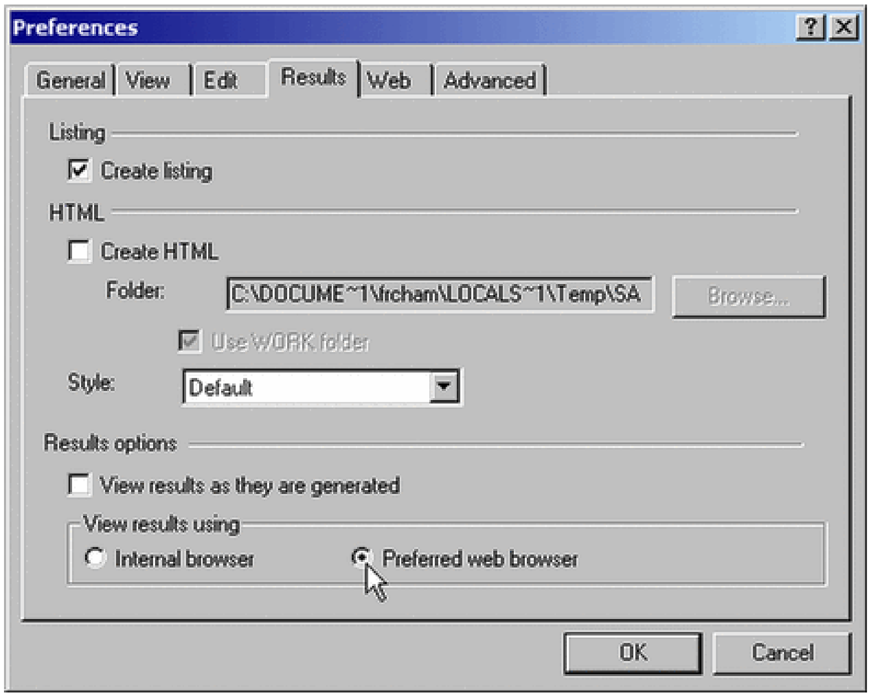
HTML
[ ] Create HTML
Folder: C:\DOCUME~1\frcham\LOCALS~1\Temp\SA
[✓] Use WORK folder
Style: Default

Results options
[ ] View results as they are generated
View results using
(●) Internal browser ( ) Preferred web browser

Figure 9.12. Disabling View of Results as Generated

*If you are using the LATEX or the PS destinations you must use a PostScript viewer, such as Ghostview.

You can change the default to use an external viewer instead of the Results Viewer. Select **Tools** → **Options** → **Preferences. . .** from the menu at the top of the main SAS window. Then in the **Results** tab select **Preferred web browser**, as shown in Figure 9.13. Your results will then be displayed in the default viewer that is configured in your computer for the corresponding destination.



Listing

[✓] Create listing

HTML

[ ] Create HTML

Folder: C:\DOCUME~1\frcham\LOCALS~1\Temp\SA

[✓] Use WORK folder

Style: Default

Results options

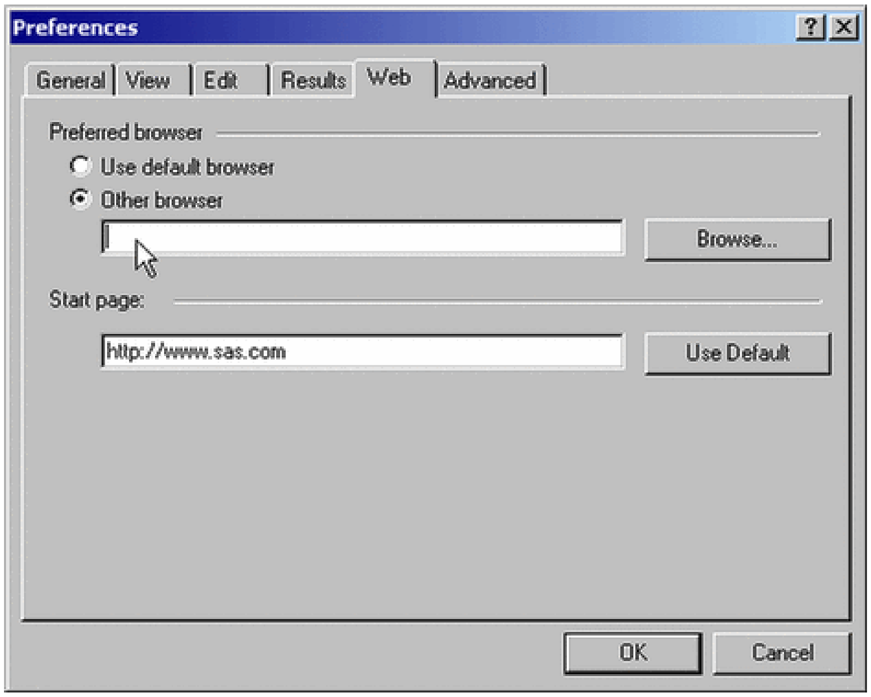
[ ] View results as they are generated

View results using

○ Internal browser ● Preferred web browser

Figure 9.13. Selecting an External Browser

You can also choose which browser to use for HTML output. Select **Tools** → **Options** → **Preferences. . .** from the menu at the top of the main SAS window. Then in the **Web** tab select **Other browser**, and type (or browse) the path of your preferred browser, as shown in Figure 9.14.



Preferred browser

( ) Use default browser

[✓ Other browser]

Start page: http://www.sas.com

Figure 9.14. Changing the Default External Browser

Referring to Graphs by Name

Procedures assign a name to each graph they create with ODS Graphics. This enables you to refer to ODS graphs in the same way that you refer to ODS tables (see the "Using the Output Delivery System" section on page 244 in Chapter 8, "Using the Output Delivery System"). You can determine the names of graphs in several ways:

- You can look up graph names in the "ODS Graphics" section of chapters for procedures that use ODS Graphics. See, for example, the "ODS Graphics" section on page 1166 in Chapter 20, "The MODEL Procedure."
- You can use the Results window to view the names of ODS graphs created in your SAS session. See the section "Using ODS with the SAS Explorer" on page 247 for more information.
- You can use the ODS TRACE ON statement to list the names of graphs created by your SAS session. This statement adds identifying information in the SAS log (or, optionally, in the SAS listing) for each graph that is produced. See page 279 for an example, and the "Using the Output Delivery System" section on page 244 for more information.

Note that the graph name is not the same as the name of the file containing the graph (see page 284).

Selecting and Excluding Graphs

You can use graph names to specify which ODS graphs are displayed with the ODS SELECT and ODS EXCLUDE statements. See the section "Using the Output Delivery System" on page 244 for information on how to use these statements.

Example

This example revisits the analysis described in the section "Using the ODS GRAPHICS Statement and Procedure Options" on page 272.

To determine which output objects are created by ODS, you specify the ODS TRACE ON statement prior to the procedure statements.

```
ods trace on;

ods html;
ods graphics on;

proc ucm data=melanoma;
   id year interval=year;
   model Melanoma_Incidence_Rate;
   irregular;
   level variance=0 noest;
   slope variance=0 noest;
   cycle rho=1 noest=rho plot=smooth;
   estimate back=5;
   forecast back=5 lead=10 plot=forecasts print=none;
run;

ods graphics off;
ods html close;
```

Figure 9.15 displays an extract from the trace record, which is added to the SAS log. By default, the UCM procedure creates several table objects and two graph objects named "SmoothedCyclePlot" and "ModelForecastsPlots." In addition to the name, the trace record provides the label, template, and path for each output object. Graph templates are distinguished from table templates by a naming convention that uses the procedure name in the second level and the word "Graphics" in the third level. For example, the fully qualified template name for the forecasts plot created by PROC UCM, as shown in Figure 9.15, is

```
Ets.UCM.Graphics.ModelForecastsPlot
```

```
Output Added:
-------------
Name:       DataSet
Label:      Input Data Set
Template:   ETS.UCM.DataSet
Path:       UCM.DataSet
-------------
.
.
.

Output Added:
-------------
Name:       Forecasts
Label:      Forecasts
Template:   ets.UCM.Forecasts
Path:       UCM.Results.Forecasts
-------------
WARNING: Statistical graphics displays created with ODS are
         experimental in this release.

Output Added:
-------------
Name:       SmoothedCyclePlot
Label:      Smoothed Cycle Component
Template:   ets.UCM.Graphics.S_Cycle
Path:       UCM.Results.SmoothedCyclePlot
-------------

Output Added:
-------------
Name:       ModelForecastsPlot
Label:      Model and Forecast Plot
Template:   ets.UCM.Graphics.ModelForecastsPlot
Path:       UCM.Results.ModelForecastsPlot
-------------
```

Figure 9.15. Extract from the ODS Trace Record in SAS Log

Note that you can specify the LISTING option in the ODS TRACE ON statement to write the trace record to the LISTING destination:

```
   ods trace on / listing;
```

The following statements use the ODS SELECT statement to specify that only the two graph objects named “Contour” and “SurfacePlot” are to be included in the HTML output.

```
   ods html;
   ods graphics on;

   ods select ModelForecastsPlot;

   proc ucm data=melanoma noprint;
      id year interval=year;
      model Melanoma_Incidence_Rate;
      irregular;
      level variance=0 noest;
      slope variance=0 noest;
      cycle rho=1 noest=rho plot=smooth;
      estimate back=5;
      forecast back=5 lead=10 plot=(forecasts) print=none;
   run;

   ods graphics off;
   ods html close;
```

A sample program named odsgrgs.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Specifying Styles for Graphics

ODS styles control the overall look of your output. A style definition provides formatting information for specific visual aspects of your SAS output. For ODS tables this information typically includes a list of font definitions (each font defines a family, size, weight, and style) and a list of colors, which are associated with common areas of printed output, including titles, footnotes, by-groups, table headers, and table cells.

Starting with SAS 9, ODS styles also include graphical appearance information such as line and marker properties in addition to font and color information. Furthermore, in SAS 9.1, ODS styles include graphics appearance informats for common elements of statistical graphics created with ODS Graphics. These elements include fitted lines, confidence and prediction bands, and outliers.

For more information about styles, refer to the "TEMPLATE Procedure: Creating a Style Definition" in the *SAS Output Delivery System User's Guide*.

Specifying a Style

You can specify a style using the STYLE= option in a valid ODS destination,* such as HTML, PDF, RTF, or PRINTER. Each style produces output with the same content, but a somewhat different visual appearance. For example, the following statement request output using the "Journal" style.

```
   ods html style = Journal;
```

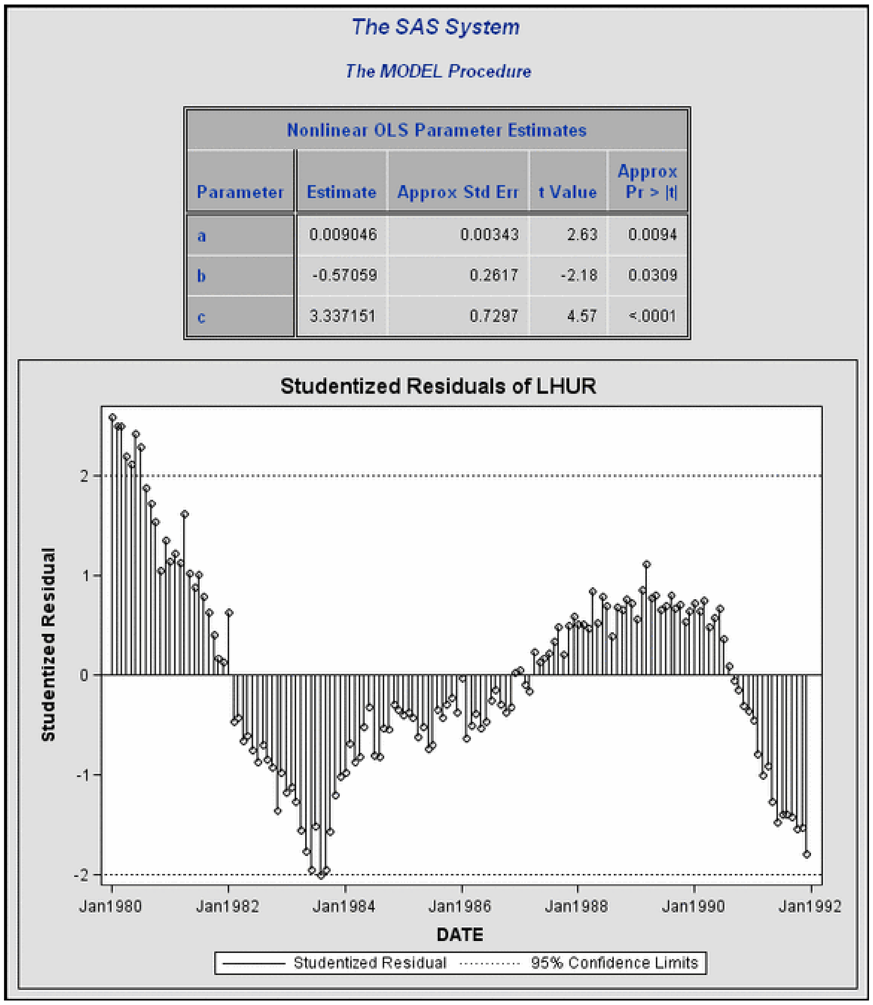
Any SAS-supplied or user-defined style can be used for ODS Graphics. However, of the SAS-supplied styles for SAS 9.1, four are specifically designed and recommended for use with ODS Graphics:

*Style definitions do not apply to the LISTING destination, which uses the SAS monospace format by default for output tables. The LISTING destination is not a valid destination for ODS Graphics in SAS 9.1.

- Analysis
- Default
- Journal
- Statistical

Figure 9.16 and Figure 9.17 illustrate the difference between the "Default" and the "Journal" styles for the HTML destination. Note that the appearance of tables and graphics is coordinated within a particular style. This is also illustrated in the series of examples starting with Example 9.11.

For more information about styles for ODS Graphics, see the section "Styles for Graphics" on page 293 or refer to the "ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)" at
http://support.sas.com/documentation/onlinedoc/base/.



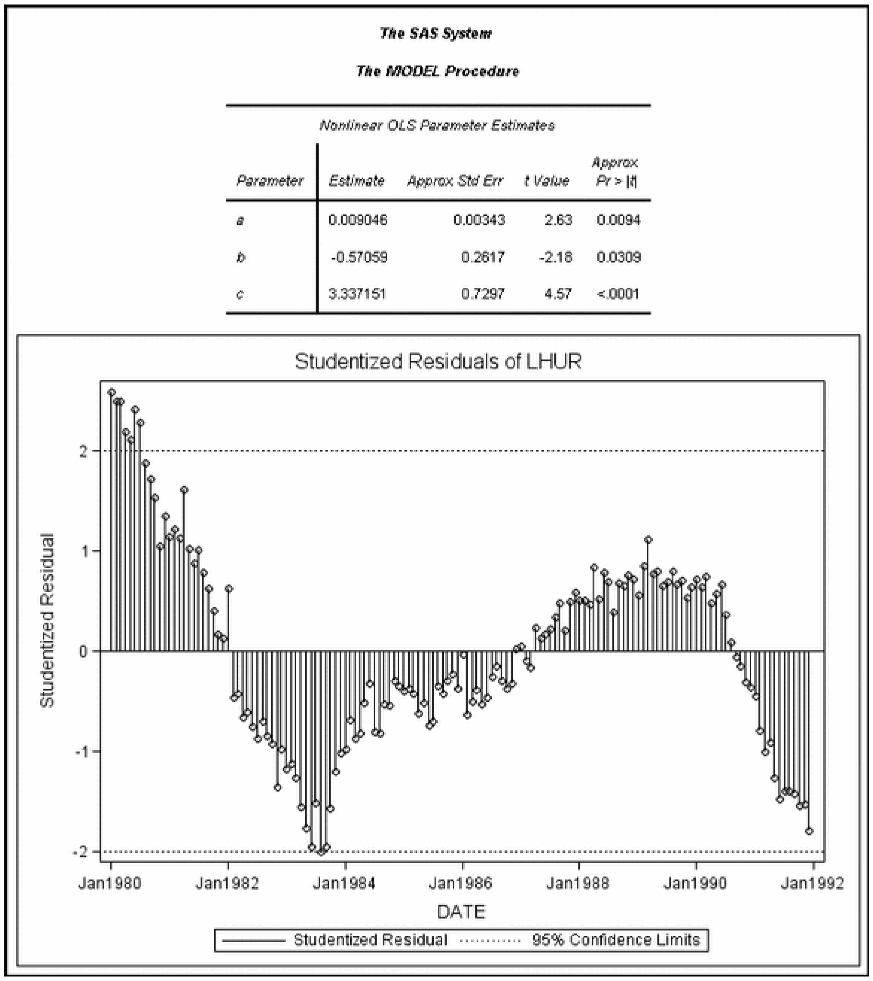
The SAS System

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	0.009046	0.00343	2.63	0.0094
b	-0.57059	0.2617	-2.18	0.0309
c	3.337151	0.7297	4.57	<.0001

Figure 9.16. HTML Output with Default Style



The SAS System

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	*Estimate*	*Approx Std Err*	*t Value*	*Approx Pr > |t|*
a	0.009046	0.00343	2.63	0.0094
b	-0.57059	0.2617	-2.18	0.0309
c	3.337151	0.7297	4.57	<.0001

Figure 9.17. HTML Output with Journal Style

Graphics Image Files

Accessing your graphs as individual image files is useful when you want to include them in various types of documents. The default image file type depends on the ODS destination, but there are other supported image file types that you can specify. You can also specify the names for your graphics image files and the directory in which you want to save them.

This section describes the image file types supported by ODS Graphics, and it explains how to name and save graphics image files.

Describing Supported Image File Types

If you are using an HTML or a LATEX destination, your graphs are individually produced in a specific image file type, such as GIF or PostScript.

If you are using a destination in the PRINTER family or the RTF destination, the graphs are contained in the ODS output file and cannot be accessed as individual image files. However, you can open an RTF output file in Microsoft Word and then copy and paste the graphs into another document, such as a Microsoft PowerPoint presentation; this is illustrated in Example 9.3.

Table 9.2 shows the various ODS destinations supported by ODS Graphics, the viewer that is appropriate for displaying graphs in each destination, and the image file types supported for each destination.

Table 9.2. Destinations and Image File Types Supported by ODS Graphics

Destination	Destination Family	Viewer	Image File Types
DOCUMENT		Not Applicable	Not Applicable
HTML	MARKUP	Browser	GIF (default), JPEG, PNG
LATEX	MARKUP	Ghostview	PostScript (default), EPSI, GIF, JPEG, PNG
PCL	PRINTER	Ghostview	Contained in PostScript file
PDF	PRINTER	Acrobat	Contained in PDF file
PS	PRINTER	Ghostview	Contained in PostScript file
RTF		Microsoft Word	Contained in RTF file

Note: In SAS 9.1 the LISTING destination does not support ODS Graphics. You must specify a supported ODS destination in order to produce ODS Graphics, as illustrated by all the examples in this chapter.

Naming Graphics Image Files

The names of graphics image files are determined by a *base file name*, an *index counter*, and an *extension*. By default, the base file name is the ODS graph name (see page 278). The index counter is set to zero when you begin a SAS session, and it is increased by one after you create a graph, independently of the graph type or the SAS procedure that creates it. The extension indicates the image file type.

For instance, if you run the example on page 272 at the beginning of a SAS session, the two graphics image files created are SmoothedCyclePlot0.gif and ModelForecastsPlot1.gif. If you immediately rerun this example, then ODS creates the same graphs in different image files named SmoothedCyclePlot2.gif and ModelForecastsPlot3.gif.

You can specify the RESET option in the ODS GRAPHICS statement to reset the index counter to zero. This is useful to avoid duplication of graphics image files if you are rerunning a SAS program in the same session.

```
   ods graphics on / reset;
```

Note: The index counter is initialized to zero at the beginning of your SAS session or if you specify the RESET option in the ODS GRAPHICS statement. Graphics image files with the same name are overwritten.

You can specify a base file name for all your graphics image files with the IMAGENAME= option in the ODS GRAPHICS statement. For example:

```
ods graphics on / imagename = "MyName";
```

You can also specify

```
ods graphics on / imagename = "MyName" reset;
```

With the preceding statement, the graphics image files are named MyName0, MyName1, and so on.

You can specify the image file type for the HTML or LATEX destinations with the IMAGEFMT= option in the ODS GRAPHICS statement. For example:

```
ods graphics on / imagefmt = png;
```

For more information, see the "ODS GRAPHICS Statement" section on page 298.

Saving Graphics Image Files

Knowing where your graphics image files are saved and how they are named is particularly important if you are running in batch mode, if you have disabled the SAS Results Viewer (see page 276), or if you plan to access the files for inclusion in a document. The following discussion assumes you are running SAS under the Windows operating system. If you are running on a different operating system, refer to the SAS Companion for your operating system.

Your graphics image files are saved by default in the SAS current folder. If you are using the SAS windowing environment, the current folder is displayed in the status line at the bottom of the main SAS window (see also page 275). If you are running your SAS programs in batch mode, the graphs are saved by default in the same directory where you started your SAS session.

For instance, suppose the SAS current folder is C:\myfiles. If you specify the ODS GRAPHICS statement, then your graphics image files are saved in the directory C:\myfiles. Unlike traditional high resolution graphics created with SAS/GRAPH, ODS Graphics are not temporarily stored in a catalog in your Work directory.

With the HTML and the LATEX destinations, you can specify a directory for saving your graphics image files. With the PRINTER and RTF destinations, you can only specify a directory for your output file. The remainder of this discussion provides details for each destination type.

HTML Destination

If you are using the HTML destination, the individual graphs are created as GIF files by default. You can use the PATH= and GPATH= options in the ODS HTML statement to specify the directory where your HTML and graphics files are saved, respectively. This also gives you more control over your graphs. For example, if you want to save your HTML file named test.htm in the C:\myfiles directory, but you want to save your graphics image files in C:\myfiles\gif, then you specify

```
ods html path  = "C:\myfiles"
         gpath = "C:\myfiles\gif"
         file  = "test.htm";
```

When you specify the URL= suboption with the GPATH= option, SAS creates relative paths for the links and references to the graphics image files in the HTML file. This is useful for building output files that are easily moved from one location to another. For example, the following statements create a relative path to the gif directory in all the links and references contained in test.htm.

```
ods html path  = "C:\myfiles"
         gpath = "C:\myfiles\gif" (url="gif/")
         file  = "test.htm";
```

If you do not specify the URL= suboption, SAS creates absolute paths that are hard-coded in the HTML file; these may cause broken links if you move the files. For more information, refer to the ODS HTML statement in the “Dictionary of ODS Language Statements” (*SAS Output Delivery System User’s Guide*).

LATEX Destination

$\LaTeX$ is a document preparation system for high-quality typesetting. The experimental ODS LATEX statement produces output in the form of a $\LaTeX$ source file that is ready to compile in $\LaTeX$.

When you request ODS Graphics for a LATEX destination, ODS creates the requested graphs as PostScript files by default, and the $\LaTeX$ source file includes references to these image graphics files. You can compile the $\LaTeX$ file or you can access the individual PostScript files to include your graphs in a different $\LaTeX$ document, such as a paper that you are writing.

You can specify the PATH= and GPATH= options in the ODS LATEX statement, as explained previously for the ODS HTML statement. See Example 9.4 for an illustration.

The ODS LATEX statement is an alias for the ODS MARKUP statement using the TAGSET=LATEX option. For more information, refer to the ODS MARKUP statement in the “Dictionary of ODS Language Statements” (*SAS Output Delivery System User’s Guide*).

If you are using a LATEX destination with the default PostScript image file type, your ODS graphs are created in gray-scale, regardless of the style you are using. When you use this destination, it is recommended that you use the "Journal" style to obtain high quality graphics. For more information about styles, see the "Specifying Styles for Graphics" section on page 281.

To create color graphics using a LATEX destination, specify JPEG, GIF, or PNG with the IMAGEFMT= option in the ODS GRAPHICS statement. If you specify GIF you can use a distiller to obtain a PostScript or a PDF file. If you specify JPEG you may need to include the `\DeclareGraphicsExtensions` and the `\DeclareGraphicsRule` commands in the preamble of your LaTeX file. For more information, refer to the LaTeX documentation for the `graphicx` package.

PRINTER and RTF Destinations

If you are using a destination in the PRINTER family (PCL, PDF, PS) or the RTF destination, the graphs are contained in the output file and cannot be accessed as individual graphics image files. You can specify the path where the output file is to be saved using the FILE= option of the ODS destination statement. For example, suppose that you specify

```
ods pdf file = "test.pdf";
```

Then your ODS output is saved as the PDF file test.pdf in the SAS current folder (for example, in C:\myfiles).

You can specify a full path name for your output with the FILE= option. For instance to save your PDF file to the directory C:\temp you specify

```
ods pdf file = "C:\temp\test.pdf";
```

You can always check the SAS log to verify where your output is saved. For example, the preceding statement would result in the following log message:

```
NOTE: Writing ODS PDF output to DISK destination
   "C:\temp\test.pdf", printer "PDF".
```

Customizing Graphics with Templates

This section describes how to locate templates for ODS Graphics, and how to display, edit, and save these templates. It also provides an overview of the graph template language. Before presenting these details, a review of the TEMPLATE procedure terminology is helpful.

A *template definition* is a set of SAS statements that can be run with the TEMPLATE procedure to create a compiled template. Two common types of template definitions are *table definitions* and *style definitions*. A table definition describes how to display the output for an output object that is to be rendered as a table, and a style definition provides formatting information for specific visual aspects of your SAS output.

A third type of template definition is a *graph template definition* (or *graph definition* for short), which controls the layout and details of graphs produced with ODS Graphics. Graph definitions begin with a DEFINE STATGRAPH statement and end with an END statement.

A *template store* is a member of a SAS data library that stores compiled templates created by the TEMPLATE procedure. Default templates supplied by SAS are saved in the Sashelp.Tmplmst template store.

In common applications of ODS Graphics, it should not be necessary to modify the default template for each graph, which is supplied by SAS. However, when customization is necessary, you can modify the default template with the graph template language in the TEMPLATE procedure.

If you are using the SAS windowing environment, the easiest way to display, edit, and save your templates is by using the Templates window. For detailed information about managing templates, refer to the "TEMPLATE Procedure: Managing Template Stores" in the *SAS Output Delivery System User's Guide*.

For details concerning the syntax of the graph template language, refer to the "TEMPLATE Procedure: Creating ODS Statistical Graphics Output (Experimental)" at http://support.sas.com/documentation/onlinedoc/base/.

Locating Templates

The first step in customizing a graph is to determine which template was used to create the original graph. The easiest way to do this is to specify the ODS TRACE ON statement prior to the procedure statements that created the graph. The fully qualified template name is displayed in the SAS log. This is illustrated in Example 9.7 and the section "Using the Output Delivery System" on page 244. Note that the ODS TRACE ON statement applies to graphs just as it does to tables.

Displaying Templates

Once you have found the fully qualified name of a template, you can display its definition using one of these methods:

- Open the Templates window by typing **odstemplates** (or **odst** for short) in the command line, as shown in Figure 9.18. If you expand the **Sashelp.Tmplmst** icon, you can browse all the available templates and double-click on any template icon to display its definition. This is illustrated in Example 9.7.

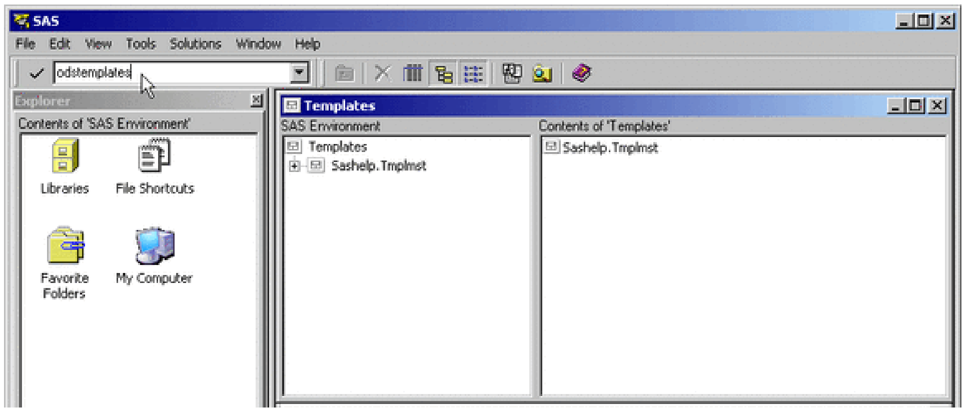


Figure 9.18. Requesting the Templates Window in the Command Line

- Use the SOURCE statement in PROC TEMPLATE to display a template definition in the SAS log. For example, the following statements display the default definition of the residual Q-Q plot in PROC MODEL.

```
proc template;
   source Ets.Model.Graphics.QQPlot;
run;
```

Editing Templates

You can modify the format and appearance of a particular graph by modifying its template. There are several ways to edit a template definition:

- Find the template icon in the Templates window, right-click on the icon, and select **Edit** from the pull-down menu. This opens a Template Editor window in which you can edit the template definition. This approach is illustrated in Example 9.7.
- Find the template icon in the Templates window and double-click on the template icon to display the template definition. Copy and paste the template definition into the Program Editor.
- Use the SOURCE statement with the FILE= option in PROC TEMPLATE. This writes the template definition to a file that you can modify. For example:

```
proc template;
   source Ets.Model.Graphics.QQPlot /
          file = "qqtpl.sas";
run;
```

By default the file is saved in the SAS current folder. Note that with this approach you have to add a PROC TEMPLATE statement before the template definition statements and a RUN statement at the end before submitting your modified definition.

Note: Graph definitions are self-contained and do not support parenting as do table definitions. For more information about graph definitions and the graph template language see the "Introducing the Template Language for Graphics" section on page 291.

Saving Customized Templates

After you edit the template definition you can submit your PROC TEMPLATE statements as you would for any other SAS program:

- If you are using the Template Editor window, select **Submit** from the **Run** menu. For example, see Example 9.7.
- Alternatively, submit your PROC TEMPLATE statements in the Program Editor.

ODS automatically saves the compiled template in the first template store that it can update, according to the currently defined ODS path. If you have not changed the ODS path, then the modified template is saved in the Sasuser.Templat template store. You can display the current ODS path with the following statement.

```
ods path show;
```

By default, the result of this statement is displayed in the SAS log, as illustrated in Figure 9.19.

```
Current ODS PATH list is:

1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

Figure 9.19. Result of ODS PATH SHOW Statement

Using Customized Templates

When you create ODS output (either graphs or tables) with a SAS program, ODS searches sequentially through each element of the ODS PATH list for the first template that matches the ODS name of each output object requested. This template is used to produce the output object. If you have not changed the default ODS path, then the first template store searched is Sasuser.Templat, followed by Sashelp.Tmplmst.

Note that you can have templates with the same name in different template stores. The template that is used is the first one found in the ODS path.

The ODS PATH statement specifies which locations to search for definitions that were created by PROC TEMPLATE, as well as the order in which to search for them. You can change the default path by specifying different locations in the ODS PATH statement. For example, the following statement changes the default ODS path so that the first template store searched is Work.Mypath.

```
ods path work.mypath(update) sashelp.tmplmst(read);
```

The UPDATE option provides update access as well as read access to Work.Mypath. The READ option provides read-only access to Sashelp.Tmplmst.

For more information, refer to the ODS PATH Statement in the “Dictionary of ODS Language Statements” (*SAS Output Delivery System User’s Guide*).

Reverting to Default Templates

Customized templates are stored in Sasuser.Templat or in user-defined template stores. The default templates provided by SAS are saved in the read-only template store Sashelp.Tmplmst. Consequently, if you have modified any of the default templates and you want to create ODS Graphics with the original default templates, one way to do so is by changing your ODS path as follows.

```
ods path sashelp.tmplmst(read) sasuser.templat(update);
```

A second approach, which is highly recommended, is to save all your customized templates in a user-defined template store (for example Work.Mypath). Then you can reset the default ODS path with the ODS PATH RESET statement:

```
ods path reset;
```

A third approach is to save your customized definition as part of your SAS program and delete the corresponding template from your Sasuser.Templat template store.

Example 9.7 illustrates all the steps of displaying, editing, saving and using customized templates.

Introducing the Template Language for Graphics

Graph template definitions are written in a *graph template language*, which has been added to the TEMPLATE procedure in SAS 9.1. This language includes statements for specifying plot layouts (such as grids or overlays), plot types (such as scatter plots and histograms), and text elements (such as titles, footnotes, and insets). It also provides support for built-in computations (such as histogram binning) and evaluation of expressions. Options are available for specifying colors, marker symbols, and other attributes of plot features.

Graph template definitions begin with a DEFINE STATGRAPH statement in PROC TEMPLATE, and they end with an END statement. You can specify the DYNAMIC statement to define dynamic variables, the MVAR and NMVAR statements to define macro variables, and the NOTES statement to provide descriptive information about the graph.

The statements available in the graph template language can be classified as follows:

- **Control statements**, which specify conditional or iterative flow of control. By default, flow of control is sequential. In other words, each statement is used in the order in which it appears.
- **Layout statements**, which specify the arrangement of the components of the graph. Layout statements are arranged in blocks which begin with a LAYOUT statement and end with an ENDLAYOUT statement. The blocks can be nested. Within a layout block, you can specify plot, text, and other statement types to define one or more graph components. Statement options provide control for attributes of layouts and components.
- **Plot statements**, which specify a number of commonly used displays, including series plots, autocorrelation plots, histograms, and scatter plots. Plot statements are always provided within a layout block. The plot statements include options to specify which data columns from the source objects are used in the graph. For example, in the SCATTERPLOT statement used to define a scatter plot, there are mandatory X= and Y= options that specify which data columns are used for the x- and y-variables in the plot, and there is a GROUP= option that specifies a data column as an optional classification variable.
- **Text statements**, which specify descriptions accompanying the graphs. An entry is any textual description, including titles, footnotes, and legends, and it can include symbols to identify graph elements.

As an illustration, the following statements display the template definition of the Series plot available in PROC TIMESERIES (see "ODS Graphics" in Chapter 28, "The TIMESERIES Procedure").

```
proc template;
   link Ets.Timeseries.Graphics.SeriesPlot to
        Statgraph.TimeSeries.SeriesPlot;
   define statgraph Statgraph.TimeSeries.SeriesPlot;
      dynamic title Time Series IntegerTime;
      layout Gridded;
         EntryTitle TITLE / padbottom=5;
         layout Overlay /
            XGrid     = True
            YGrid     = True
            XAxisOpts = ( Integer = INTEGERTIME );
            SeriesPlot x = TIME y = SERIES /
               markers      = true
               markercolor  = GraphDataDefault:contrast
               markersymbol = GraphDataDefault:markersymbol
               markersize   = GraphDataDefault:markersize
               linecolor    = StatGraphDataLine:contrastcolor
               linepattern  = StatGraphFitLine:linestyle;
         EndLayout;
      EndLayout;
   end;
run;
```

The DEFINE STATGRAPH statement in PROC TEMPLATE creates the graph template definition. The DYNAMIC statement defines dynamic variables. The variable

TITLE provides the title of the graph. The variables TIME and SERIES contain the time variable and the time series. The variable INTEGERTIME is a binary variable that can assume a value of TRUE or FALSE depending on whether an ID statement is specified in the procedure. You can use these dynamic text variables in any text element of the graph definition.

The overall display is specified with the LAYOUT GRIDDED statement. The title of the graph is specified with the ENTRYTITLE statement inside a layout overlay block, which is nested within the main layout. The main plot is a series plot specified with the SERIESPLOT statement. The options in the SERIESPLOT statement, which are given after the slash, specify the color, symbol,and size for the markers, and color and pattern for the lines using indirect references to style attributes of the form ***style-element:attribute***. The values of these attributes are specified in the definition of the style you are using, and so they are automatically set to different values if you specify a different style. For more information about style references see the "Styles for Graphics" section on page 293.

The second ENDLAYOUT statement ends the main layout block and the END statement ends the graph template definition.

Note: Graph template definitions are self-contained and do not support parenting (inheritance) as do table definitions. The EDIT statement is not supported.

For details concerning the syntax of the graph template language, refer to the "TEMPLATE Procedure: Creating ODS Statistical Graphics Output (Experimental)" at http://support.sas.com/documentation/onlinedoc/base/.

Styles for Graphics

This section provides an overview of the style elements for ODS Graphics. It also describes how to customize a style definition and how to specify a default style for all your output.

Introducing Style Elements for Graphics

An ODS style definition is composed of a set of *style elements*. A style element is a collection of *style attributes* that apply to a particular feature or aspect of the output. A value is specified for each attribute in a style definition.

Style definitions control the overall appearance of ODS tables and graphs. For ODS tables, style definitions specify features such as background color, table borders, and color scheme, and they specify the fonts, sizes, and color for the text and values in a table and its headers. For ODS graphs, style definitions specify the following features:

- background color
- graph dimensions (height and width). See Example 9.13 for an illustration.
- borders
- line styles for axes and grid lines

- fonts, sizes, and colors for titles, footnotes, axis labels, axis values, and data labels. See Example 9.11 for an illustration.
- marker symbols, colors, and sizes for data points and outliers
- line styles for needles
- line and curve styles for fitted models and predicted values. See Example 9.12 for an illustration.
- line and curve styles for confidence and prediction limits
- fill colors for histogram bars, confidence bands, and confidence ellipses
- colors for box plot features
- colors for surfaces
- color ramps for contour plots

In the templates supplied by SAS for ODS graphs, options for plot features are always specified with a style reference of the form ***style-element:attribute*** rather than a hard-coded value. For example, the symbol, color, and size of markers for basic series plots are specified in a template SERIESPLOT statement as follows:

```
SeriesPlot x=TIME y=SERIES /
   markercolor  = GraphDataDefault:contrast
   markersymbol = GraphDataDefault:markersymbol
   markersize   = GraphDataDefault:markersize;
```

This guarantees a common appearance for markers used in all basic series plots, which is controlled by the **GraphDataDefault** element of the style definition that you are using.

In general, the ODS graph features listed above are determined by style element attributes unless they are overridden by a statement or option in the graph template.

In order to create your own style definition or to modify a style definition for use with ODS Graphics, you need to understand the relationships between style elements and graph features. This information is provided in the section “ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)” at http://support.sas.com/documentation/onlinedoc/base/.

Style definitions are created and modified with the TEMPLATE procedure. For more information, refer to the “TEMPLATE Procedure: Creating a Style Definition” in the *SAS Output Delivery System User’s Guide*.

Customizing Style Definitions

The default style definitions that SAS provides are stored in the “Styles” directory of Sashelp.Tmplmst.

You can display, edit, and save style definitions using the same methods available for modifying template definitions, as explained in the sections beginning on page 288. In particular, you can display style definitions using one of these methods:

- If you are using the Templates window in the SAS windowing environment, expand the **Sashelp.Tmplmst** node under **Templates**, and then select **Styles** to display the contents of this folder.
- Use the SOURCE statement in PROC TEMPLATE. For example, the following statements display the "Journal" style definition in the SAS log.

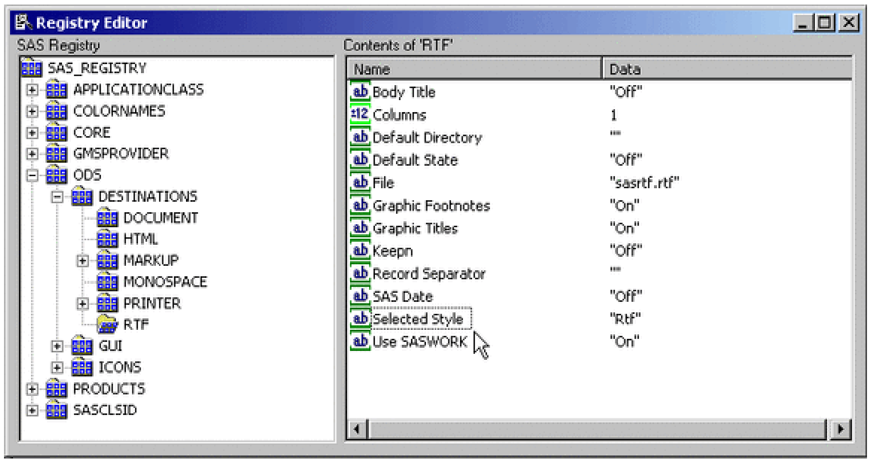
```
proc template;
   source Styles.Journal;
run;
```

Specifying a Default Style

The default style for each ODS destination is specified in the SAS Registry. For example, the default style for the HTML destination is "Default," and for the RTF destination it is "Rtf."

You can specify a default style for all your output in a particular ODS destination. This is useful if you want to use a different SAS-supplied style, if you have modified one of the SAS-supplied styles (see page 294), or if you have defined your own style. For example, you can specify the "Journal" style for all your RTF output.

The recommended approach for specifying a default style is as follows. Open the SAS Registry Editor by typing **regedit** in the command line. Expand the node **ODS** → **DESTINATIONS** and select a destination (for example, select **RTF**). Double-click the **Selected Style** item, as illustrated in Figure 9.20, and specify a style. This can be any SAS-supplied style or a user-defined style, as long as it can be found with the current ODS path (for example, specify **Journal**). You can specify a default style for the HTML, MARKUP, and PRINTER destinations in a similar way.



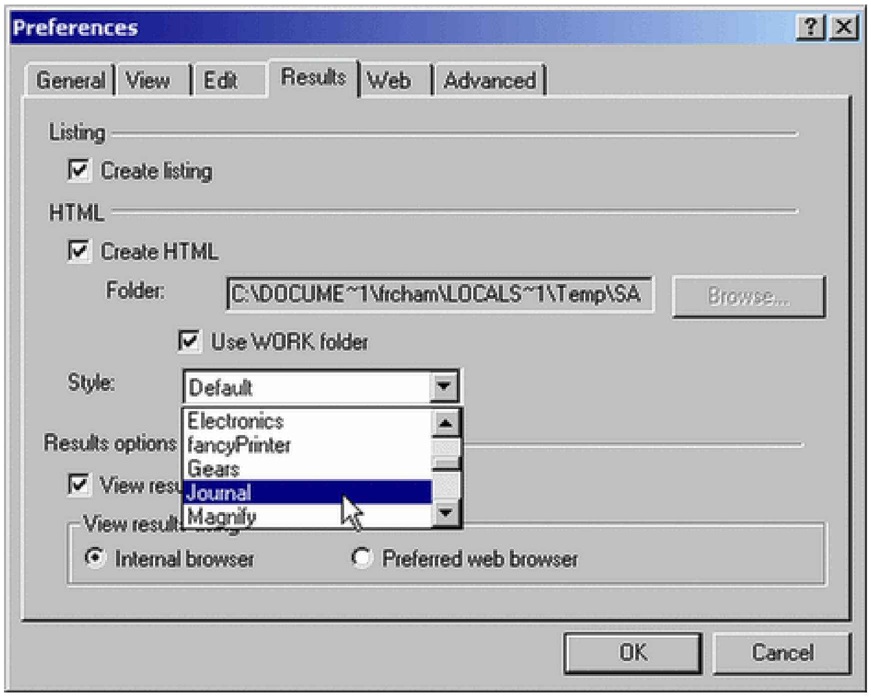
Name	Data
Body Title	"Off"
Columns	1
Default Directory	""
Default State	"Off"
File	"sasrtf.rtf"
Graphic Footnotes	"On"
Graphic Titles	"On"
Keepn	"Off"
Record Separator	""
SAS Date	"Off"
Selected Style	"Rtf"
Use SASWORK	"On"

Figure 9.20. SAS Registry Editor

Note: ODS searches sequentially through each element of the ODS PATH list for the first style definition that matches the name of the style specified in the SAS Registry. The first style definition found is used. If you are specifying a customized style as your default style, the following are useful suggestions:

- If you save your style in Sasuser.Templat, verify that the name of your default style matches the name of the style specified in the SAS Registry. For example suppose the “Rtf” style is specified for the RTF destination in the SAS Registry. You can name your style Rtf and save it in Sasuser.Templat. This blocks the “Rtf” style in Sashelp.Tmplmst.
- If you save your style in a user-defined template store, verify that this template store is the first in the current ODS PATH list. Include the ODS PATH statement in your SAS autoexec file so that it is executed at startup.

For the HTML destination, an alternative approach for specifying a default style is as follows. From the menu at the top of the main SAS window select **Tools** → **Options** → **Preferences. . . .** In the **Results** tab check the **Create HTML** box and select a style from the pull-down menu. This is illustrated in Figure 9.21.


Preferences

General | View | Edit | Results | Web | Advanced

Listing
☑ Create listing

HTML
☑ Create HTML
Folder: C:\DOCUME~1\frcham\LOCALS~1\Temp\SA
☑ Use WORK folder
Style: Default

Electronics
fancyPrinter
Gears
Journal
Magnify

Results options
☑ View resu

View result
(●) Internal browser ( ) Preferred web browser

Figure 9.21. Selecting a Default Style for HTML Destination

Details

Procedures Supporting ODS Graphics

The following SAS procedures support ODS Graphics in SAS 9.1:

Base SAS

- CORR

SAS/ETS

- ARIMA
- AUTOREG
- ENTROPY
- EXPAND
- MODEL
- SYSLIN
- TIMESERIES
- UCM
- VARMAX
- X12

SAS High-Performance Forecasting

- HPF

SAS/STAT

- ANOVA
- CORRESP
- GAM
- GENMOD
- GLM
- KDE
- LIFETEST
- LOESS
- LOGISTIC
- MI
- MIXED
- PHREG
- PRINCOMP
- PRINQUAL
- REG
- ROBUSTREG

For details on the specific graphs available with a particular procedure, see the “ODS Graphics” section in the corresponding procedure chapter.

Operating Environments Supporting ODS Graphics

The following operating systems are supported:

- Windows (32- and 64- bit)
- OpenVMS Alpha
- z/OS (OS/390)
- UNIX (AIX, HP-UX, Tru64 UNIX, Solaris, Linux)

For information specific to an operating system, refer to the SAS Companion for that operating system.

Creating ODS Graphics in z/OS

Creating ODS Graphics with the z/OS (OS/390) operating system requires the following to be configured by your System Administrator:

- Java
- UNIX File System components

For more information, refer to the sections “Installing UNIX File System Components” and “Configuring SAS Software for Use with the Java Platform” of the *SAS System Configuration Guide*.

In addition, when you specify an ODS HTML destination you must specify the PATH= or GPATH= option with a valid UNIX directory.

ODS GRAPHICS Statement

The basic syntax for enabling ODS Graphics is

```
ods graphics on;
```

You specify this statement prior to your procedure statements, as illustrated in the “Using the ODS GRAPHICS Statement” section on page 267. Any procedure that supports ODS Graphics then produces graphics, either by default or when you specify procedure options for requesting particular graphs.

To disable ODS Graphics, specify

```
ods graphics off;
```

The following is a summary of the ODS GRAPHICS statement syntax. You can find the complete syntax in the section ODS Graphics Statement in the “Dictionary of ODS Language Statements” (*SAS Output Delivery System User’s Guide*).

Syntax

ODS GRAPHICS < **OFF | ON** < / *options* > > **;**

enables ODS to create graphics automatically. The default is ON.

Options

ANTIALIAS | NOANTIALIAS
ANTIALIAS = ON | OFF

controls the use of antialiasing to smooth the components of a graph.

OFF

suppresses the use of antialiasing for components other than text.

ON

specifies that antialiasing is to be used to smooth jagged edges of all of the components in a graph.

Text displayed in a graph is always antialiased. If the number of observations in the ODS output object exceeds 250, then antialiasing is not used, even if you specify the option ANTIALIAS=ON. The default is ON.

IMAGEFMT = < *image-file-type* **| STATIC | STATICMAP** >

specifies the image file type (directly or indirectly) for displaying graphics in ODS output. The default image file type depends on the ODS destination; it is used when you specify IMAGEFMT=STATIC. You can also specify other supported image file types. This option only applies to ODS Graphics, and it has no effect on traditional high resolution graphics that rely on GOPTIONS values. The default is STATIC.

image-file-type

specifies the type of image you want to add to your graph. If the image file type is not valid for the active output destination, the default is used instead. Table 9.3 lists the image file types supported for the ODS destinations that are valid with ODS Graphics.

STATIC

specifies the best quality image file type for the active output destination.

STATICMAP

applies only with the HTML destination and specifies that an HTML image map is to be created for tool tip support. The image file type used is the same as with STATIC. For an illustration see Example 9.2. If the number of observations in the data set exceeds 500, the image map is not generated.

Table 9.3. Supported Destinations and Image File Types

Destination	**Values for IMAGEFMT= Option**
HTML	GIF (default), JPEG, PNG
LATEX	PS (default), EPSI, GIF, JPEG, PNG
PCL	Not applicable
PDF	Not applicable
PS	Not applicable
RTF	Not applicable

Note: For PCL, PDF, PS, and RTF, the IMAGEFMT= option is not applicable because the graph is contained in the output file. See Table 9.2.

IMAGENAME = <*file-name*>

specifies the base image file name. The default is the name of the output object. You can determine the name of the output object by using the ODS TRACE statement. The base image name should not include an extension. ODS automatically adds the increment value and the appropriate extension (which is specific to the output destination that has been selected).

RESET

resets the index counter appended to image file names.

Note: The index counter is initialized to zero at the beginning of your SAS session or if you specify the RESET option in the ODS GRAPHICS statement. Graphics image files with the same name are overwritten.

Examples

This section provides a series of examples which illustrate various tasks that can be performed with ODS Graphics. The examples are presented in increasing order of task complexity and should be read sequentially.

Example 9.1. Selecting and Excluding Graphs

This example illustrates how to select and exclude ODS graphs from your output.

The “Getting Started” example on page 267 uses the MODEL procedure to produce the plots shown in Figure 9.1 through Figure 9.8.

The ODS TRACE ON statement requests a record of the output objects created by ODS, which is displayed in the SAS log as shown in Output 9.1.1.

```
ods trace on;

ods html;
ods graphics on;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;

ods graphics off;
ods html close;
```

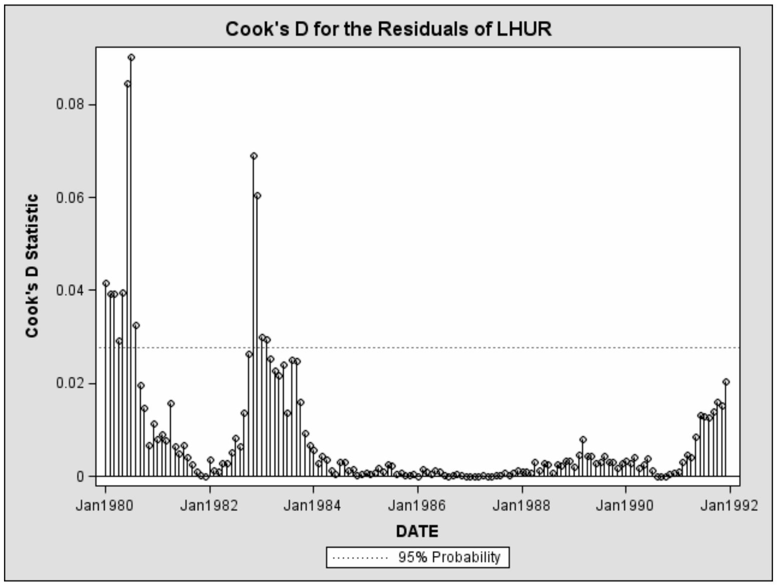
Output 9.1.1. Partial ODS Trace Record in SAS Log

```
Output Added:
-------------
Name:       ModSummary
Label:      Variable Counts
Template:   ets.model.ModSummary
Path:       Model.ModSum.ModSummary
-------------

.
.
.


Output Added:
-------------
Name:       EstSummaryStats
Label:      Estimation Summary Statistics
Template:   ets.model.EstSummaryStats
Path:       Model.OLS.EstSummaryStats
-------------
WARNING: Statistical graphics displays created with ODS are
         experimental in this release.

Output Added:
-------------
Name:       StudentResidualPlot
Label:      Studentized Residuals of LHUR
Template:   ETS.Model.Graphics.StudentResidualPlot
Path:       Model.OLS.StudentResidualPlot
-------------

Output Added:
-------------
Name:       CooksD
Label:      Cook’s D for the Residuals of LHUR
Template:   ETS.Model.Graphics.CooksD
Path:       Model.OLS.CooksD
-------------

.
.
.

Output Added:
-------------
Name:       ResidualHistogram
Label:      Histogram of Residuals of LHUR
Template:   ETS.Model.Graphics.ResidualHistogram
Path:       Model.OLS.ResidualHistogram
-------------
```

You can use the ODS SELECT statement to restrict your output to a particular subset of ODS tables or graphs. The following statements restrict the output to the Cook’s D plot, which is shown in Output 9.1.2.

```
ods html;
ods graphics on;


proc model data=sashelp.citimon;
   ods select CooksD;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;

ods graphics off;
ods html close;
```

Output 9.1.2. Cook's D Plot



Conversely, you can use the ODS EXCLUDE statement to display all the output with the exception of a particular subset of tables or graphs. For example, to exclude the studentized residuals plot from the output you specify

```
ods exclude StudentResidualPlot;
```

See the "Selecting and Excluding Graphs" section on page 279 for further information.

A sample program named odsgr01.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.2. Creating Graphs with Tool Tips in HTML

This example demonstrates how to request graphics in HTML with tool tip displays, which appear when you move a mouse over certain features of the graph. When you specify the HTML destination and the IMAGEFMT=STATICMAP option in the ODS GRAPHICS statement, then the HTML file output file is generated with an image map of coordinates for tool tips. The individual graphs are saved as GIF files.

Example 28.3 of Chapter 28, “The TIMESERIES Procedure” utilizes the SASHELP.WORKERS data set to study the time series of electrical workers and its interaction with the series of masonry workers.

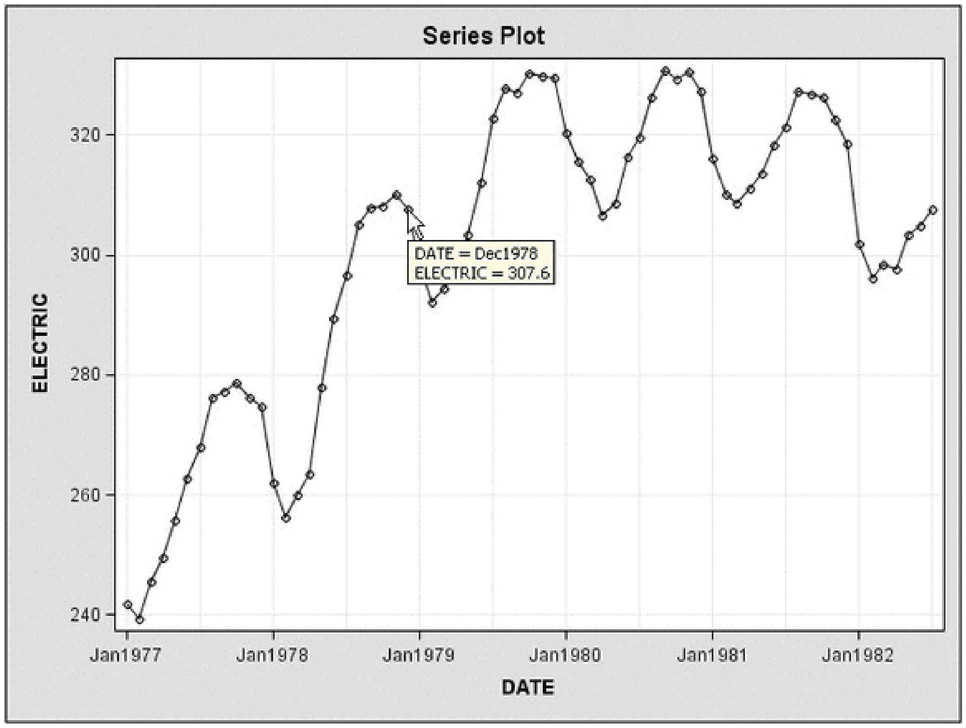
The following statements request a plot of the series of electrical workers using the PLOT option in the PROC TIMESERIES statement.

```
ods html;
ods graphics on / imagefmt = staticmap;

proc timeseries data=sashelp.workers out=_null_
   plot=series;
   id date interval=month;
   var electric;
   crossvar masonry;
run;

ods graphics off;
ods html close;
```

Output 9.2.1 displays the series plot that is included in the HTML output.

Moving the mouse over a data point shows a tool tip with the corresponding identifying information.

Output 9.2.1. Series Plot with Tool Tips



Note: Graphics with tool tips are only supported for the HTML destination.

A sample program named odsgr02.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.3. Creating Graphs for a Presentation

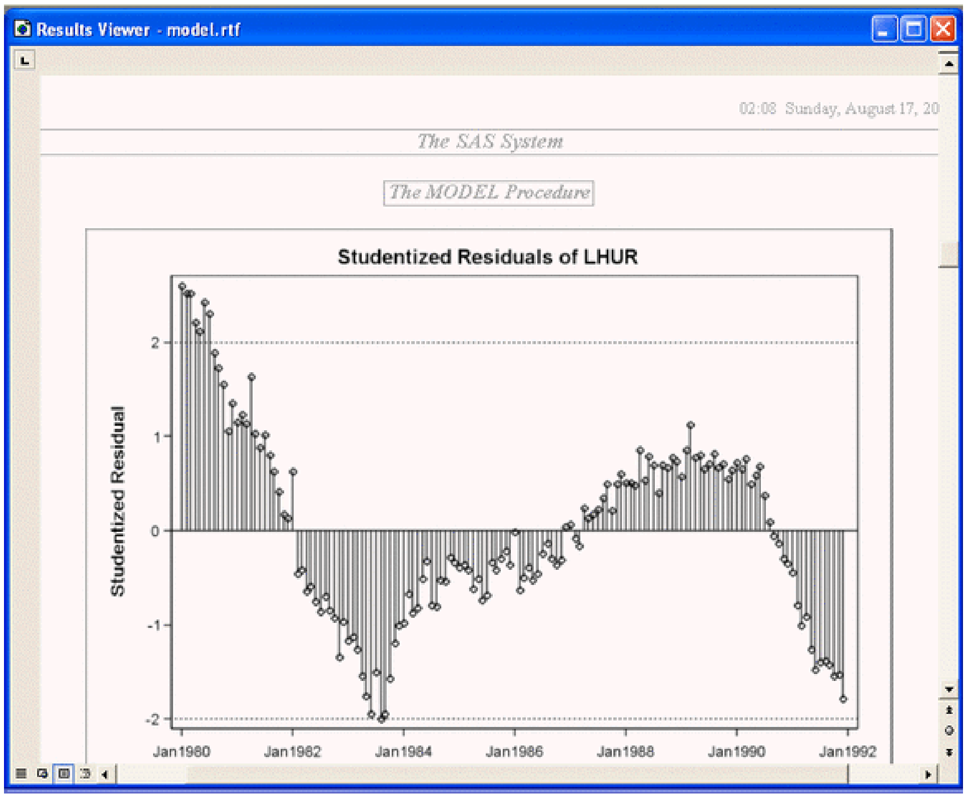
The RTF destination provides the easiest way to create ODS graphs for inclusion in a document or presentation. You can specify the ODS RTF statement to create a file that is easily imported into a word processor (such as Microsoft Word or WordPerfect) or a presentation (such as Microsoft PowerPoint).

In this example, the following statements request that the output of Example 9.1 be saved in the file model.rtf.

```
ods rtf file = "model.rtf";
ods graphics on;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;

ods graphics off;
ods rtf close;
```

The output file includes various tables and the following plots: a studentized residual plot, a Cook's D plot, a predicted by actual values plot, an autocorrelation of residuals, a partial autocorrelation of residuals, an inverse autocorrelation of residuals, a QQ plot of residuals, and a histogram of the residuals. The studentized residuals plot is shown in Output 9.3.1.

Output 9.3.1. Studentized Residuals Plot



If you are running SAS in the Windows operating system, it is easy to include your graphs in a Microsoft PowerPoint presentation when you generate RTF output. You can open the RTF file in Microsoft Word and simply copy and paste the graphs into Microsoft PowerPoint. In general, RTF output is convenient for exchange of graphical results between Windows applications through the clipboard.

Alternatively, if you request ODS Graphics using the HTML destination, then your individual graphs are created as GIF files by default. You can insert the GIF files into a Microsoft PowerPoint presentation. See "Naming Graphics Image Files" and "Saving Graphics Image Files" for information on how the image files are named and saved.

A sample program named odsgr03.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.4. Creating Graphs in PostScript Files

This example illustrates how to create individual graphs in PostScript files, which is particularly useful when you want to include them in a LaTeX document.

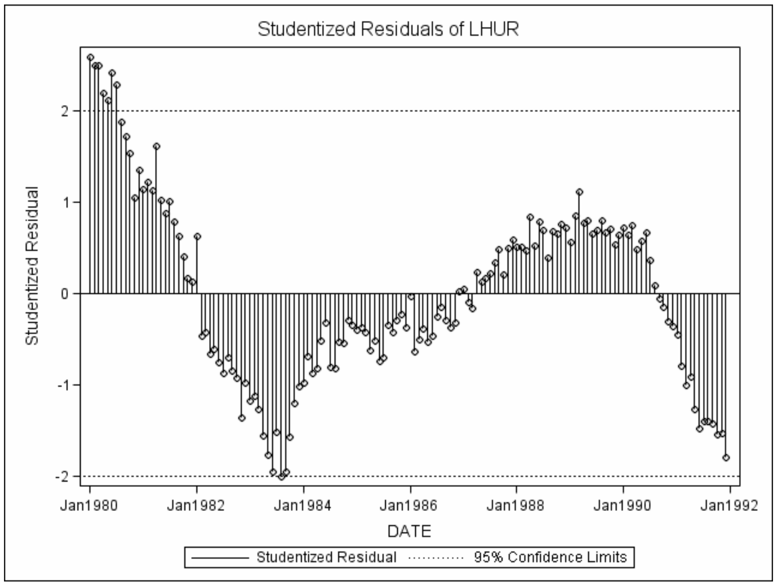
The following statements specify a LATEX destination[*] for the output in Example 9.3 with the “Journal” style.

```
ods latex style=Journal;
ods graphics on;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;

ods graphics off;
ods latex close;
```

The “Journal” style displays gray-scale graphs that are suitable for a journal. When you specify the ODS LATEX destination, ODS creates a PostScript file for each individual graph in addition to a LaTeX source file that includes the tabular output and references to the PostScript files. By default these files are saved in the SAS current folder. If you run this example at the beginning of your SAS session, the studentized residual plot shown in Output 9.4.1 is saved by default in a file named StudentResidualPlot0.ps. See page 284 for details about how graphics image files are named.

[*]The LATEX destination in ODS is experimental in SAS 9.1.

Output 9.4.1. Histogram Using Journal Style



If you are writing a paper, you can include the graphs in your own LaTeX source file by referencing the names of the individual PostScript graphics files. In this situation, you may not find necessary to use the LaTeX source file created by SAS.

If you specify PATH= and GPATH= options in the ODS LATEX statement, your tabular output is saved as a LaTeX source file in the directory specified with the PATH= option, and your graphs are saved as PostScript files in the directory specified with the GPATH= option. This is illustrated by the following statements:

```
ods latex path  = "C:\temp"
          gpath = "C:\temp\ps" (url="ps/")
          style = Journal;
ods graphics on;

   ...SAS statements...

ods graphics off;
ods latex close;
```

The URL= suboption is specified in the GPATH= option to create relative paths for graphs referenced in the LaTeX source file created by SAS. See the "HTML Destination" section on page 286 for further information.

A sample program named odsgr04.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.5. Creating Graphs in Multiple Destinations

This example illustrates how to send your output to more than one destination with a single execution of your SAS statements.

For instance, to create both HTML and RTF output, you can specify the ODS HTML and the ODS RTF statements before your procedure statements.

```
ods html;
ods rtf;

   ...SAS statements...

ods _all_ close;
```

The ODS _ALL_ CLOSE statement closes all open destinations.

You can also specify multiple instances of the same destination. For example, using the data in the "Using the ODS GRAPHICS Statement and Procedure Options" section on page 272, the following statements save the smoothed cycle component plot to the file smoothcycle.pdf and the forecasts plot to the file forecasts.pdf.

```
ods pdf file="smoothcycle.pdf";
ods pdf select SmoothedCyclePlot;

ods pdf(id=frcst) file="forecasts.pdf";
ods pdf(id=frcst) select ModelForecastsPlot;

ods graphics on;

proc ucm data=melanoma noprint;
   id year interval=year;
   model Melanoma_Incidence_Rate;
   irregular;
   level variance=0 noest;
   slope variance=0 noest;
   cycle rho=1 noest=rho plot=smooth;
   estimate back=5;
   forecast back=5 lead=10 plot=forecasts print=none;
run;

ods graphics off;
ods _all_ close;
```

The ID= option assigns the name srf to the second instance of the PDF destination. Without the ID= option, the second ODS PDF statement would close the destination that was opened by the previous ODS PDF statement, and it would open a new instance of the PDF destination. In that case, the file smoothcycle.pdf would contain no output. For more information, refer to the Example 1 of the ODS PDF statement in the "Dictionary of ODS Language Statements" (*SAS Output Delivery System User's Guide*).

Example 9.6. Displaying Graphs Using the DOCUMENT Procedure

This example illustrates the use of the DOCUMENT destination and the DOCUMENT procedure to display your ODS graphs. In particular, this is useful when you want to display your output (both tables and graphs) in one or more ODS destinations, or when you want to use different styles without rerunning your SAS program.

In general, when you send your output to the DOCUMENT destination you can use the DOCUMENT procedure to rearrange, duplicate, or remove output from the results of a procedure or a database query. You can also generate output for one or more ODS destinations. For more information, refer to the ODS DOCUMENT statement in the “Dictionary of ODS Language Statements” and “The DOCUMENT Procedure” (*SAS Output Delivery System User’s Guide*).

The following statements repeat the estimation of the model in Example 9.1. The ODS DOCUMENT statement stores the data for the tables and the plots from this analysis in an ODS document named lhurDoc. Neither the tables nor the plots are displayed.

```
ods listing close;
ods document name=lhurDoc(write);
ods graphics on;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;
quit;

ods graphics off;
ods document close;
ods listing;
```

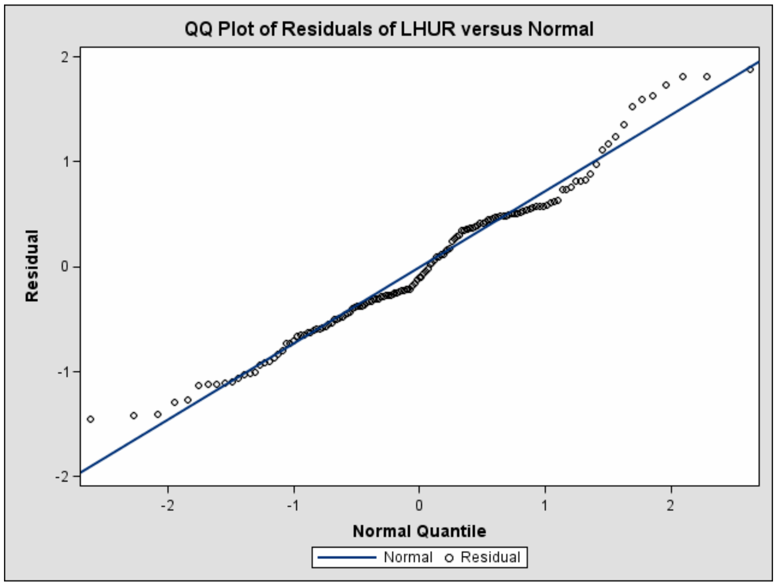
If you want to display, for example, the Q-Q plot of residuals using PROC DOCUMENT, you first need to determine its name. You can do this by specifying the ODS TRACE ON statement prior to the procedure statements (see page 278 for more information). Alternatively, you can type **odsdocuments** (or **odsd** for short) in the command line to open the Documents window, which you can then use to manage your ODS documents.

The following statements specify an HTML destination and display the residual Q-Q plot using the REPLAY statement in PROC DOCUMENT.

```
ods html;

ods select QQPlot;

proc document name = lhurDoc;
   replay;
run;
quit;

ods html close;
```

By default, the REPLAY statement attempts to replay every output object stored in the document, but only the Q-Q plot is displayed as specified by the ODS SELECT statement. The plot is displayed in Output 9.6.1.

Output 9.6.1. Q-Q Plot Displayed by PROC DOCUMENT



As an alternative to running PROC DOCUMENT with an ODS SELECT statement, you can run PROC DOCUMENT specifying a *document path* for the Q-Q plot in the REPLAY statement. This approach is preferable when the document contains a large volume of output, because PROC DOCUMENT does not attempt to process every piece of output stored in the document.

You can determine the document path for the Q-Q plot by specifying the LIST statement with the LEVELS=ALL option in PROC DOCUMENT.

```
proc document name = lhurDoc;
   list / levels = all;
run;
quit;
```

This lists the entries of the QQDoc document, as shown in Output 9.6.2.

Output 9.6.2. Contents of lhurDoc

```
Listing of: \Work.Lhurdoc\
Order by: Insertion
Number of levels: All

  Obs    Path                                                                Type
--------------------------------------------------------------------------------
      1 \Model#1                                                             Dir
      2 \Model#1\ModSum#1                                                    Dir
      3 \Model#1\ModSum#1\ModSummary#1                                       Table
      4 \Model#1\ModSum#1\ModVars#1                                          Tree
      5 \Model#1\ModSum#1\Equations#1                                        Tree
      6 \Model#1\OLS#1                                                       Dir
      7 \Model#1\OLS#1\ConvergenceStatus#1                                   Table
      8 \Model#1\OLS#1\EstSum#1                                              Dir
      9 \Model#1\OLS#1\EstSum#1\DatasetOptions#1                             Table
     10 \Model#1\OLS#1\EstSum#1\MinSummary#1                                 Table
     11 \Model#1\OLS#1\EstSum#1\ConvCrit#1                                   Table
     12 \Model#1\OLS#1\EstSum#1\ObsUsed#1                                    Table
     13 \Model#1\OLS#1\ResidSummary#1                                        Table
     14 \Model#1\OLS#1\ParameterEstimates#1                                  Table
     15 \Model#1\OLS#1\EstSummaryStats#1                                     Table
     16 \Model#1\OLS#1\StudentResidualPlot#1                                 Graph
     17 \Model#1\OLS#1\CooksD#1                                              Graph
     18 \Model#1\OLS#1\ActualByPredicted#1                                   Graph
     19 \Model#1\OLS#1\ACFPlot#1                                             Graph
     20 \Model#1\OLS#1\PACFPlot#1                                            Graph
     21 \Model#1\OLS#1\IACFPlot#1                                            Graph
     22 \Model#1\OLS#1\QQPlot#1                                              Graph
     23 \Model#1\OLS#1\ResidualHistogram#1                                   Graph
```

The document path of the “QQPlot” entry in lhurDoc, as shown in Output 9.6.2, is

```
\Model#1\OLS#1\QQPlot#1
```

You can specify this path to display the residual Q-Q plot with PROC DOCUMENT as follows.

```
ods html;

proc document name = lhurDoc;
   replay \Model#1\OLS#1\QQPlot#1;
run;
quit;

ods html close;
```

You can also determine the document path from the Results window or the Documents window. Right-click on the object icon and select **Properties**.

A sample program named odsgr06.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.7. Customizing Graph Titles and Axes Labels

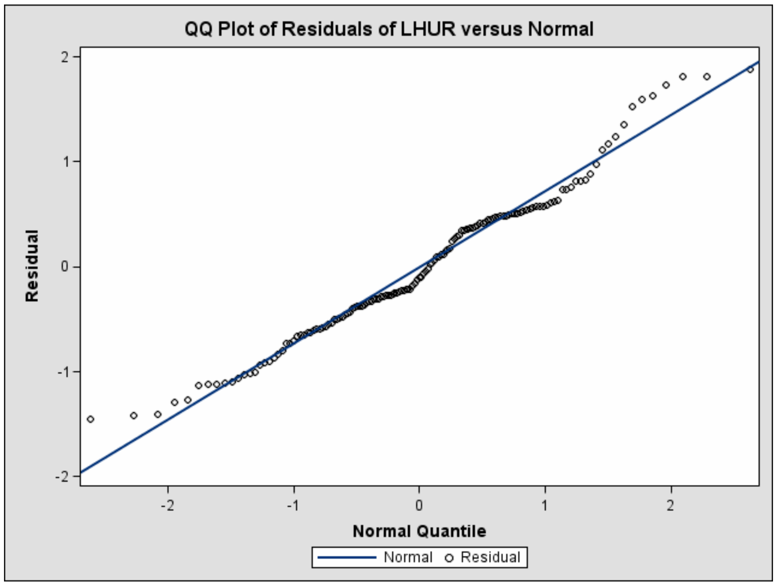
This example shows how to use PROC TEMPLATE to customize the appearance and content of an ODS graph. It illustrates the discussion in the section "Customizing Graphics with Templates" on page 287 in the context of changing the default title and y-axis label for a Q-Q plot created with the MODEL procedure.

The following statements request a Q-Q plot for residuals using PROC MODEL with the LHUR series in the library SASHELP.CITIMON for the model in Example 9.1.

```
ods trace on;
ods html;
ods graphics on;

ods select QQPlot;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;
quit;

ods graphics off;
ods html close;
ods trace off;
```

The Q-Q plot is shown in Output 9.7.1.

Output 9.7.1. Default Q-Q Plot from PROC MODEL



The ODS TRACE ON statement requests a record of all the ODS output objects created by PROC MODEL. A partial listing of the trace record, which is displayed in the SAS log, is shown in Output 9.7.2.

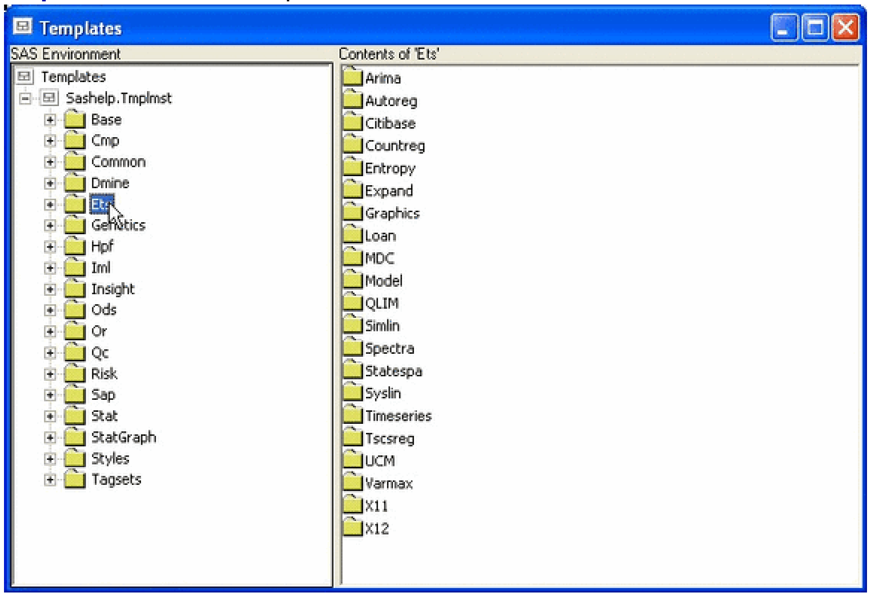
Output 9.7.2. Trace Record for Q-Q Plot

```
Output Added:
-------------
Name:       QQPlot
Label:      QQ Plot of Residuals of LHUR versus Normal
Template:   ETS.Model.Graphics.QQPlot
Path:       Model.OLS.QQPlot
-------------
```

As shown in Output 9.7.2, ODS Graphics creates the Q-Q plot using an ODS output data object named “QQPlot” and a graph template named “Ets.Model.Graphics.QQPlot,” which is the default template provided by SAS. Default templates supplied by SAS are saved in the Sashelp.Tmplmst template store (see page 287).

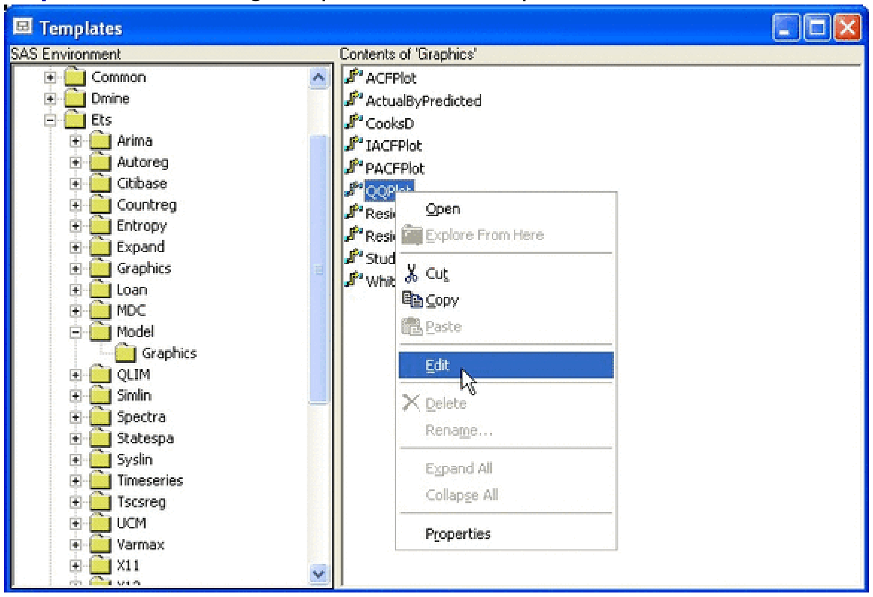
To display the default template definition, open the Templates window by typing **odstemplates** (or **odst** for short) in the command line. Expand **Sashelp.Tmplmst** and click on the **Ets** folder, as shown in Output 9.7.3.

Output 9.7.3. The Templates Window



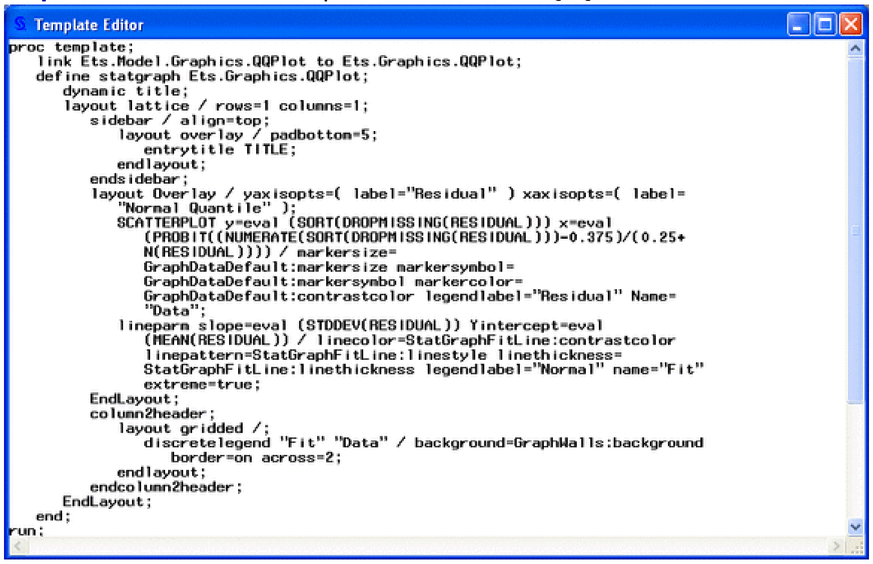
Next, open the **Model** folder and then open the **Graphics** folder. Then right-click on the “QQPlot” template icon and select **Edit**, as shown in Output 9.7.4.

Output 9.7.4. Editing Templates in the Template Window



Selecting **Edit** opens a Template Editor window, as shown in Output 9.7.5. You can use this window to edit the template.

Output 9.7.5. Default Template Definition for Q-Q Plot



```
proc template;
   link Ets.Model.Graphics.QQPlot to Ets.Graphics.QQPlot;
   define statgraph Ets.Graphics.QQPlot;
      dynamic title;
      layout lattice / rows=1 columns=1;
         sidebar / align=top;
            layout overlay / padbottom=5;
               entrytitle TITLE;
            endlayout;
         endsidebar;
         layout Overlay / yaxisopts=( label="Residual" ) xaxisopts=( label=
            "Normal Quantile" );
            SCATTERPLOT y=eval (SORT(DROPMISSING(RESIDUAL))) x=eval
               (PROBIT((NUMERATE(SORT(DROPMISSING(RESIDUAL)))-0.375)/(0.25+
               N(RESIDUAL)))) / markersize=
               GraphDataDefault:markersize markersymbol=
               GraphDataDefault:markersymbol markercolor=
               GraphDataDefault:contrastcolor legendlabel="Residual" Name=
               "Data";
            lineparm slope=eval (STDDEV(RESIDUAL)) Yintercept=eval
               (MEAN(RESIDUAL)) / linecolor=StatGraphFitLine:contrastcolor
               linepattern=StatGraphFitLine:linestyle linethickness=
               StatGraphFitLine:linethickness legendlabel="Normal" name="Fit"
               extreme=true;
         EndLayout;
         column2header;
            layout gridded /;
               discretelegend "Fit" "Data" / background=GraphWalls:background
                  border=on across=2;
            endlayout;
         endcolumn2header;
      EndLayout;
   end;
run;
```

The template definition in Output 9.7.5 is discussed below and in subsequent examples. It is listed in a more convenient format by the following statements:

```
proc template;
   link Ets.Model.Graphics.QQPlot to Ets.Graphics.QQPlot;
   define statgraph Ets.Graphics.QQPlot;
   dynamic title;
   layout lattice / rows = 1 columns = 1;
      sidebar / align=top;
         layout overlay / padbottom=5;
            entrytitle Title;
         endlayout;
      endsidebar;
      layout Overlay /
            yaxisopts = ( label = "Residuals" )
            xaxisopts = ( label = "Normal Quantile" );
         SCATTERPLOT
            y = eval(SORT(DROPMISSING(RESIDUAL)))
            x = eval(PROBIT((NUMERATE(SORT(DROPMISSING(RESIDUAL)))-0.375)/
                    (0.25+N(RESIDUAL)))) /
            markersize   = GraphDataDefault:markersize
            markersymbol = GraphDataDefault:markersymbol
            markercolor  = GraphDataDefault:contrastcolor
            legendlabel  = "Residual"
            Name         = "Data";
         lineparm
            slope      = eval(STDDEV(RESIDUAL))
            Yintercept = eval(MEAN(RESIDUAL)) /
               linecolor     = StatGraphFitLine:contrastcolor
               linepattern   = StatGraphFitLine:linestyle
               linethickness = StatGraphFitLine:linethickness
               legendlabel   = "Normal"
               name          = "Fit"
               extreme       = true;
      EndLayout;
      column2header;
         layout Gridded /;
            DiscreteLegend "Fit" "Data" /
               background = GraphWalls:background
               border     = on
               across     = 2;
         EndLayout;
      endcolumn2header;
   EndLayout;
end;
run;
```

As an alternative to using the Template Editor window, you can submit the following statements, which display the “Plot” template definition in the SAS log.

```
proc template;
   source Ets.Model.Graphics.QQPlot;
run;
```

The SOURCE statement specifies the fully qualified template name. You can copy and paste the template source into the Program Editor, modify it, and submit it using PROC TEMPLATE. See the “Editing Templates” section on page 289 for more information.

In the template, the default title of the Q-Q plot is specified by the ENTRYTITLE statement. Note that TITLE is a dynamic text variable whose values is passed by the MODEL procedure, and is QQ Plot of Residual vs Normal in Output 9.7.1). The default label for the y-axis is specified by the LABEL= suboption of the YAXISOPTS= option for the LAYOUT OVERLAY statement.

Suppose you want to change the default title to My Favorite Title, and you want the y-axis label to be Residuals of LHUR. First, replace the two ENTRYTITLE statements with the single statement

```
ENTRYTITLE "My Favorite Title" / padbottom = 5;
```

The PADBOTTOM= option specifies the amount of empty space (in pixel units) at the bottom of the layout component. In this case it creates an empty space of 5 pixels between the title and the adjacent layout component, which defines the plot itself.

Next, replace the LABEL= suboption with the following:

```
label = "Residuals of LHUR"
```

Note that you can reuse dynamic text variables such as Title in any text element.

You can then submit the modified template definition as you would any SAS program, for example, by selecting **Submit** from the **Run** menu.

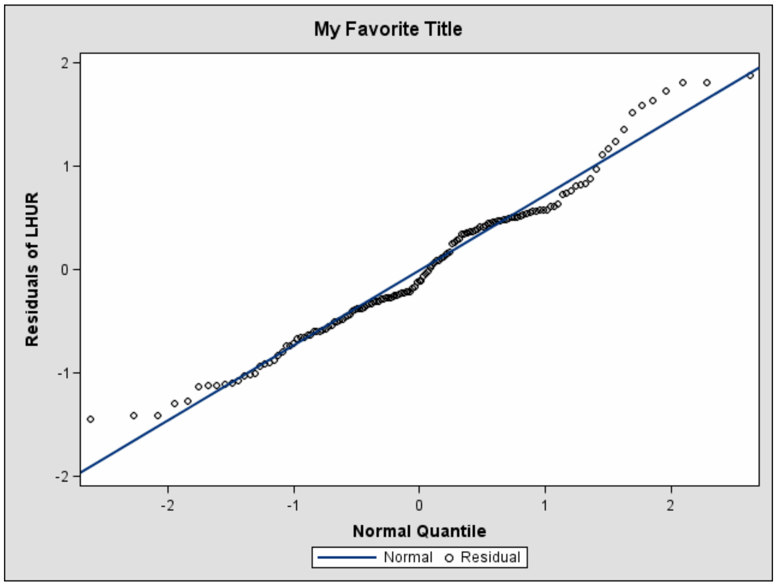
After submitting the PROC TEMPLATE statements you should see the following message in the SAS log:

```
NOTE: STATGRAPH ’Ets.Model.Graphics.QQPlot’ has been
  saved to: SASUSER.TEMPLAT
```

Note: Graph definitions are self-contained and do not support parenting as do table definitions. For more information about graph definitions and the graph template language, see the “Introducing the Template Language for Graphics” section on page 291.

Finally, resubmit the PROC MODEL statements on page 312 to display the Q-Q plot created with your modified template, as shown in Output 9.7.6.

Output 9.7.6. Q-Q Plot with Modified Title and Y-Axis Label



If you have not changed the default ODS path, the modified template “QQplot” is used automatically because Sasuser.Templat occurs before Sashelp.Tmplmst in the ODS search path. See the “Using Customized Templates” section on page 290 for additional information.

Note that you do not need to rerun the PROC MODEL analysis after you modify a graph template. After you modify your template, you can submit the PROC DOCUMENT statements in Example 9.6 to replay the Q-Q plot with the modified template.

See the “Reverting to Default Templates” section on page 291 for information on how to revert to the default template.

A sample program named odsgr07.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.8. Modifying Colors, Line Styles, and Markers

This example is a continuation of Example 9.7. Here the objective is to customize colors, line attributes, and marker symbol attributes by modifying the graph template.

In the “QQPlot” template definition shown in Output 9.7.5, the SCATTERPLOT statement specifies a scatter plot of normal quantiles versus ordered standardized residuals. The default marker symbol in the scatter plot is specified by the MARKERSYMBOL= option of the SCATTERPLOT statement:

```
   markersymbol = GraphDataDefault:markersymbol
```

The default value is a reference to the style attribute **`markersymbol`** of the style element **`GraphDataDefault`**. See the "Introducing Style Elements for Graphics" section on page 293 for more information. The actual value of the marker symbol depends on the style that you are using. In this case, since the "Default" style is used, the value of the marker symbol is Circle.

You can specify a filled circle as the marker symbol by modifying the value of the MARKERSYMBOL= option as follows.

```
markersymbol = CircleFilled
```

Note that the value of the option can be any valid marker symbol or a reference to a style attribute of the form ***`style-element:attribute`***. It is recommended that you use style attributes since these are chosen to provide consistency and appropriate emphasis based on display principles for statistical graphics. If you specify values directly in a template, you are overriding the style and run the risk of creating a graph that is inconsistent with the style definition.

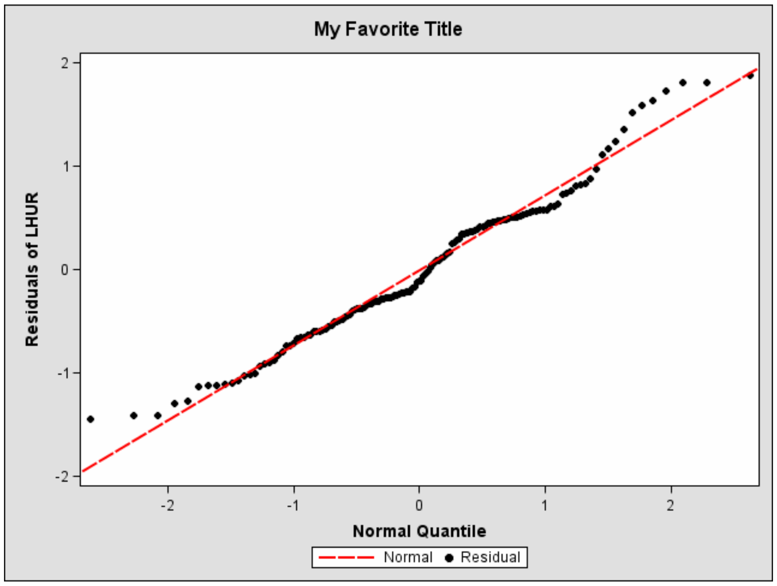
For more information about the syntax of the graphics template language and style elements for graphics, refer to the sections "TEMPLATE Procedure: Creating ODS Statistical Graphics Output (Experimental)" and "ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)" at http://support.sas.com/documentation/onlinedoc/base/.

Similarly, you can change the line color and pattern with the LINECOLOR= and LINEPATTERN= options in the LINEPARM statement. The LINEPARM statement displays a straight line specified by slope and intercept parameters. The following statements change the default color of the Q-Q plot line to red, and the line pattern to dashed.

```
linecolor   = red
linepattern = dash
```

To display these modifications, shown in Output 9.8.1, submit the modified template definition and then resubmit the PROC MODEL statements on page 312. Alternatively, you can replay the plot using PROC DOCUMENT, as in Example 9.6.

Output 9.8.1. Q-Q Plot with Modified Marker Symbols and Line



A sample program named odsgr08.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.9. Swapping the Axes in a Graph

Sometimes a Q-Q plot is displayed with the normal quantiles plotted along the y-axis and the ordered variable values plotted along the x-axis. This example, which is a continuation of Example 9.7 and Example 9.8, illustrates how to interchange the axes with a simple modification of the graph template.

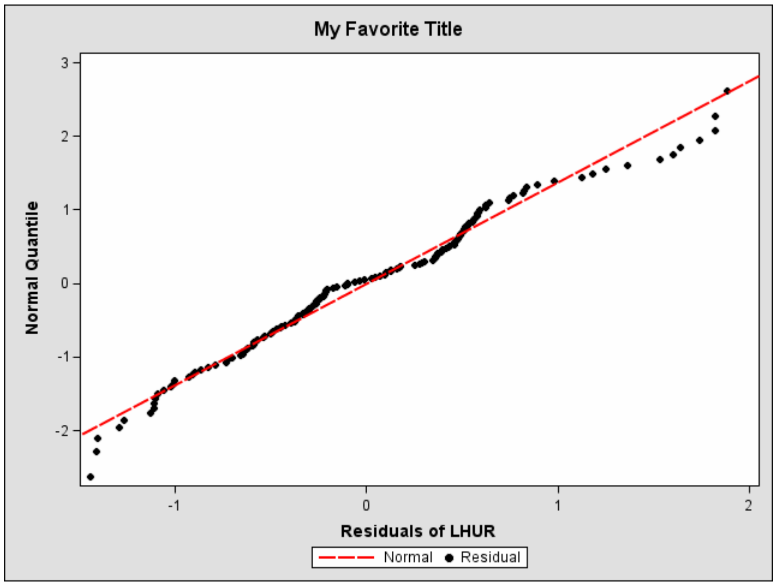
Begin by swapping the YAXISOPTS= and XAXISOPTS= options, and by swapping the X= and Y= options in the SCATTERPLOT statement.

Next, modify the LINEPARM statement. In Output 9.8.1, the slope of the line in the Q-Q plot is $\hat{\sigma}$, and y-intercept is $\hat{\mu}$. When you swap the axes, the values of the slope and y-intercept become $1/\hat{\sigma}$ and $-\hat{\mu}/\hat{\sigma}$, respectively. The modified template definition (including the changes from Example 9.7 and Example 9.8) is as follows:

```
proc template;
   link Ets.Model.Graphics.QQPlot to Ets.Graphics.QQPlot;
   define statgraph Ets.Graphics.QQPlot;
   dynamic title;
   layout lattice / rows = 1 columns = 1;
      sidebar / align = top;
         layout overlay / padbottom = 5;
            entrytitle "My Favorite Title" / padbottom=5;
         endlayout;
      endsidebar;
      layout Overlay /
             xaxisopts = ( label = "Residuals of LHUR" )
             yaxisopts = ( label = "Normal Quantile" );
         SCATTERPLOT
             x = eval(SORT(DROPMISSING(RESIDUAL)))
             y = eval(PROBIT((NUMERATE(SORT(DROPMISSING(RESIDUAL)))-0.375)/
                     (0.25+N(RESIDUAL)))) /
             markersize   = GraphDataDefault:markersize
             markersymbol = CircleFilled
             markercolor  = GraphDataDefault:contrastcolor
             legendlabel  = "Residual"
             Name         = "Data";
         lineparm
             slope      = eval(1/STDDEV(RESIDUAL))
             Yintercept = eval(-MEAN(RESIDUAL)/STDDEV(RESIDUAL)) /
                linecolor     = red
                linepattern   = dash
                linethickness = StatGraphFitLine:linethickness
                legendlabel   = "Normal"
                name          = "Fit"
                extreme       = true;
      EndLayout;
      column2header;
         layout Gridded /;
            DiscreteLegend "Fit" "Data" /
               background = GraphWalls:background
               border     = on
               across     = 2;
         EndLayout;
      endcolumn2header;
   EndLayout;
end;
run;
```

The resulting Q-Q plot, after submitting the preceding statements and the PROC MODEL statements on page 312, is shown in Output 9.9.1.

Output 9.9.1. Q-Q Plot with Swapped Axes



A sample program named odsgr09.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.10. Modifying Tick Marks and Adding Grid Lines

This example, which is a continuation of Example 9.7, Example 9.8, and Example 9.9, illustrates how to modify the tick marks for an axis and suppress grid lines.

You can use the TICKS= suboption in the XAXISOPTS= or YAXISOPTS= options to specify the tick marks for an axis. For example, you can specify the following to request tick marks ranging from -3 to 3 in the y-axis for the Q-Q plots in Output 9.9.1:

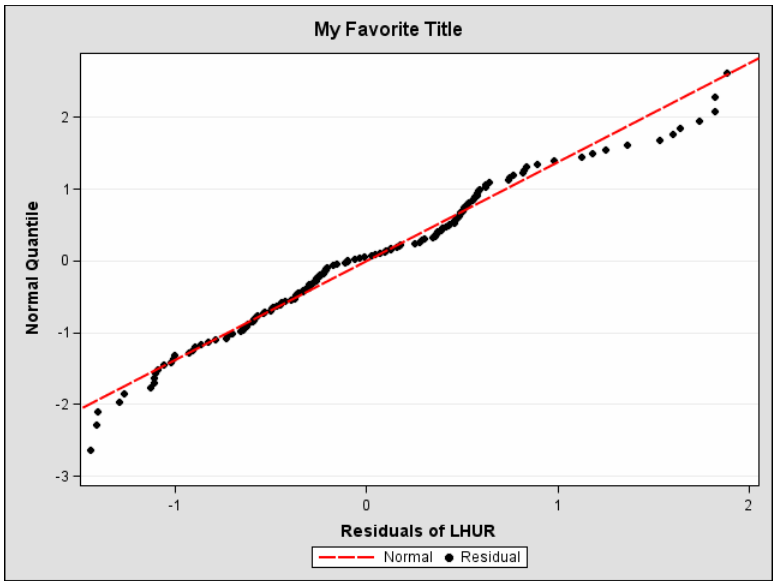
```
   yaxisopts = (label = "Normal Quantile"
                ticks = (-3 -2 -1 0 1 2))
```

By default, the Q-Q plot in Output 9.9.1 does not display grid lines. You can request vertical grid lines only by specifying

```
   XGrid = False YGrid = True
```

The result of these changes, after submitting the modified template definition and the corresponding PROC MODEL statements on page 312, is displayed in Output 9.10.1.

Output 9.10.1. Q-Q Plot with Modified Y-Axis Tick Marks and Grids



A sample program named odsgr10.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.11. Modifying Graph Fonts in Styles

You can modify an ODS style to customize the general appearance of ODS Graphics, just as you can modify a style to customize the general appearance of ODS tables. The goal of this example is to customize the fonts used in ODS graphs. It is a continuation of Example 9.10.

The following statements define a style named NewStyle that replaces the graph fonts in the “Default” style with italic Times New Roman fonts.

```
proc template;
   define style Styles.NewStyle;
   parent = Styles.Default;
   replace GraphFonts
      "Fonts used in graph styles" /
      'GraphDataFont'     = ("Times New Roman",8pt,Italic)
      'GraphValueFont'    = ("Times New Roman",10pt,Italic)
      'GraphLabelFont'    = ("Times New Roman",12pt,Italic)
      'GraphFootnoteFont' = ("Times New Roman",12pt,Italic)
      'GraphTitleFont'    = ("Times New Roman",14pt,Italic Bold);
end;
run;
```

In general, the following graph fonts are specified in the ODS styles provided by SAS:

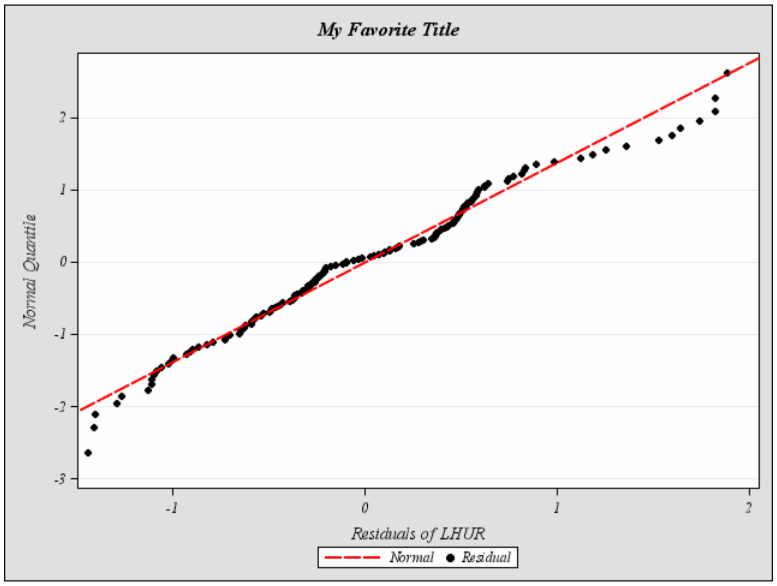
- **`'GraphDataFont'`** is the smallest font. It is used for text that needs to be small (labels for points in scatter plots, labels for contours, and so on)
- **`'GraphValueFont'`** is the next largest font. It is used for axis value (tick marks) labels and legend entry labels.
- **`'GraphLabelFont'`** is the next largest font. It is used for axis labels and legend titles.
- **`'GraphFootnoteFont'`** is the next largest font. It is used for all footnotes.
- **`'GraphTitleFont'`** is the largest font. It is used for all titles.

For more information about the DEFINE, PARENT, and REPLACE statements, refer to the "TEMPLATE Procedure: Creating a Style Definition" in the *SAS Output Delivery System User's Guide*.

The Q-Q plots in the preceding examples, beginning with Example 9.6, were created with the "Default" style; see, for instance, Output 9.10.1. In contrast, the Q-Q plot displayed in Output 9.11.1 was produced by specifying the NewStyle style in the following statements.

```
ods html style = NewStyle;
ods graphics on;

ods select QQPlot;

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;
quit;

ods graphics off;
ods html close;
```

Output 9.11.1. Q-Q Plot Using NewStyle



Although this example illustrates the use of a style with output from a particular procedure, note that a style is applied to *all* of your output (graphs and tables) in the destination for which you specify the style. See the "Specifying a Default Style" section on page 295 for information about specifying a default style for all your output.

A sample program named odsgr11.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.12. Modifying Other Graph Elements in Styles

This example, which is a continuation of Example 9.11, illustrates how to modify additional style elements for graphics, such as the thickness of a line.

The attributes of fitted lines in ODS Graphics are controlled by the style element **StatGraphFitLine**, which is defined in the "Default" style. For example, the line thickness of the normal distribution reference line in Output 9.11.1 is specified in the graph template by

```
linethickness = StatGraphFitLine:linethickness
```

To specify a line thickness of 4 pixels for the line, add the following statements to the definition of the NewStyle style in Example 9.11.

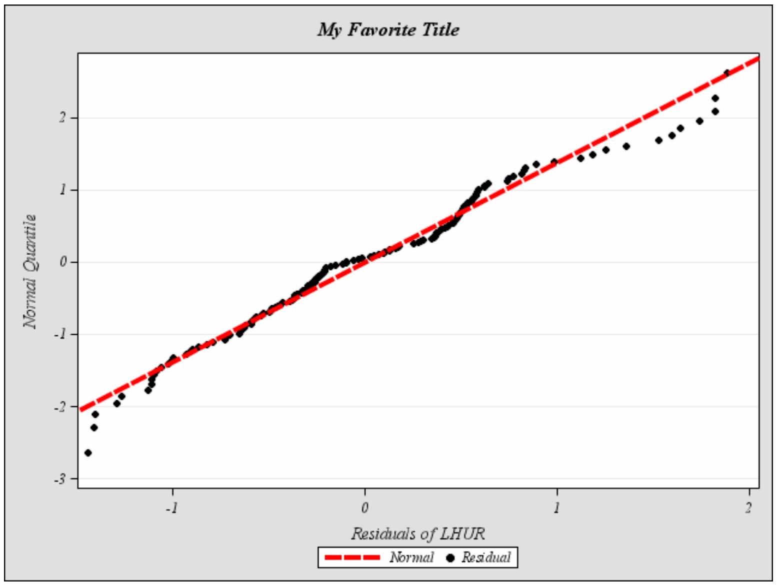
```
replace StatGraphFitLine /
   linethickness = 4px;
```

The complete revised NewStyle style is now defined by the following statements:

```
proc template;
   define style Styles.NewStyle;
   parent = Styles.Default;
   replace GraphFonts
      "Fonts used in graph styles" /
      'GraphDataFont'     = ("Times New Roman",8pt,Italic)
      'GraphValueFont'    = ("Times New Roman",10pt,Italic)
      'GraphLabelFont'    = ("Times New Roman",12pt,Italic)
      'GraphFootnoteFont' = ("Times New Roman",12pt,Italic)
      'GraphTitleFont'    = ("Times New Roman",14pt,Italic Bold);
   replace StatGraphFitLine /
      linethickness = 4px;
end;
run;
```

Output 9.12.1 shows the Q-Q plot created by the MODEL statements on page 323 with the new version of NewStyle.

Output 9.12.1. Q-Q Plot Using NewStyle with Thicker Line



You can use this approach to modify other attributes of the line, such as **`transparency`**, **`linestyle`**, **`contrastcolor`**, and **`foreground`**.

Note: Values specified directly in a graph template override style attributes. If you have customized a template, changes in a style may not have any effect. For more information, refer to the "ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)" at http://support.sas.com/documentation/onlinedoc/base/.

A sample program named odsgr12.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 9.13. Modifying Graph Sizes Using Styles

This example demonstrates how to modify the size of your ODS graphs using a style definition.

You can specify the size of a graph in a graph template definition or in a style definition:

- To modify the size of a *particular* graph, specify the dimensions with the HEIGHT= and WIDTH= options in the outermost layout of the graph template definition.
- To modify the size of *all* your ODS graphs, specify the dimensions with the OUTPUTHEIGHT= and OUTPUTWIDTH= options in the style definition.

Dimensions specified in a graph template override those specified in a style.

Continuing the discussion in Example 9.12, you can add the following style element to the definition of NewStyle to change the size of all your graphs:

```
style Graph from Graph /
   outputwidth  = 400px
   outputheight = 300px;
```

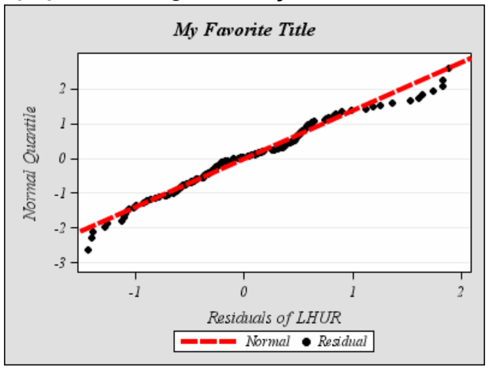
With all the changes introduced so far, NewStyle is defined as follows:

```
proc template;
   define style Styles.NewStyle;
   parent = Styles.Default;
   replace GraphFonts
      "Fonts used in graph styles" /
      'GraphDataFont'     = ("Times New Roman",8pt,Italic)
      'GraphValueFont'    = ("Times New Roman",10pt,Italic)
      'GraphLabelFont'    = ("Times New Roman",12pt,Italic)
      'GraphFootnoteFont' = ("Times New Roman",12pt,Italic)
      'GraphTitleFont'    = ("Times New Roman",14pt,Italic Bold);
   replace StatGraphFitLine /
      linethickness = 4px;
   style Graph from Graph /
      outputwidth  = 400px
      outputheight = 300px;
end;
run;
```

The dimensions of the graph must be specified in pixels. The actual size of the graph in inches depends on your printer or display device. For example, if the resolution of your printer is 100 dpi (100 dots per inch) and you want a graph that is 4 inches wide, you should set the width to 400 pixels.

You can create a smaller version of Output 9.12.1, shown in Output 9.13.1, by specifying the preceding PROC TEMPLATE statements followed by the MODEL statements on page 323.

Output 9.13.1. Q-Q Plot Using NewStyle with Smaller Dimensions



An alternative method for including smaller graphs in a document is to start with a style provided by SAS and define a modified style that *increases* the size of the graph fonts while preserving the default width and height attributes. Then you can include the graph in a document (for example in Microsoft Word) and manually rescale the graph to a smaller size while maintaining the fonts in a size that is still readable.*

The following style increases the size of the fonts but retains all the other style elements as assigned in the “Default” style:

```
proc template;
   define style Styles.BigFontStyle;
   parent = Styles.Default;
   replace GraphFonts
      "Fonts used in graph styles" /
      'GraphDataFont'     = ("Arial",12pt)
      'GraphValueFont'    = ("Arial",15pt)
      'GraphLabelFont'    = ("Arial",18pt)
      'GraphFootnoteFont' = ("Arial",18pt)
      'GraphTitleFont'    = ("Arial",21pt);
end;
run;
```

A sample program named odsgr13.sas is available for this example in the SAS Sample Library for SAS/ETS software.

*In a markup language, such as HTML or LaTeX, you can use a resize command.

Example 9.14. Modifying Panels

This example is taken from the “Getting Started” section of Chapter 61, “The REG Procedure” (*SAS/STAT User’s Guide*). It illustrates how to modify the regression fit diagnostics panel whose annotated version is shown in Output 9.14.1 so that it displays a subset of component plots. The original panel consists of eight plots and a summary statistics box. These components are labeled 1 to 9 in Output 9.14.1.

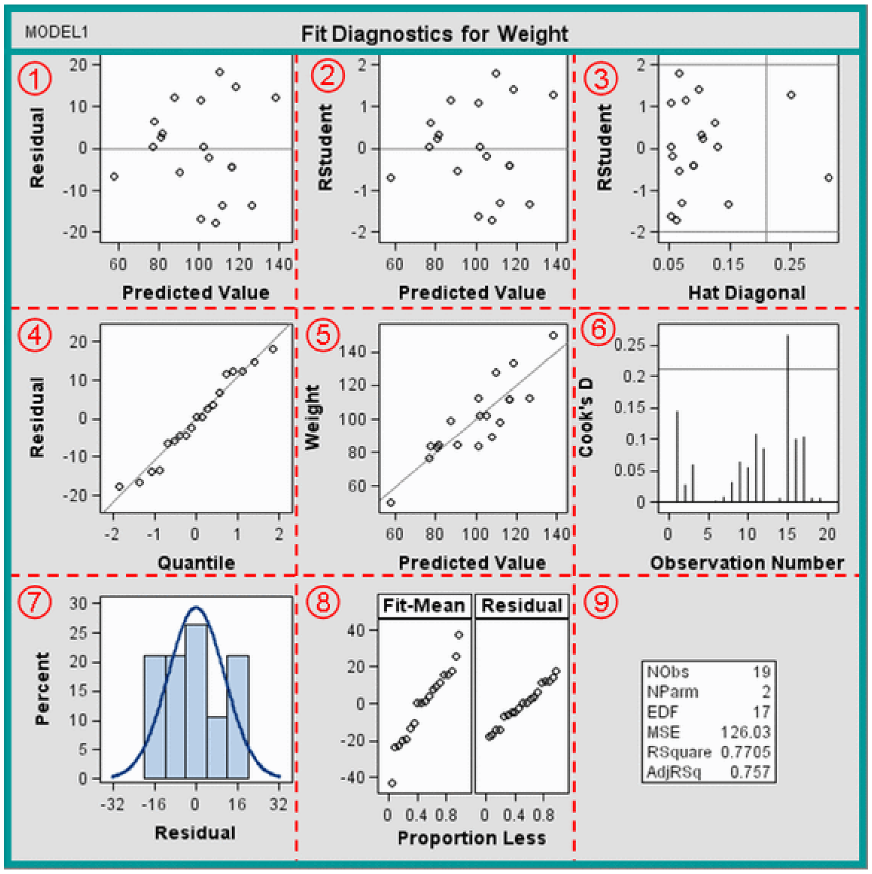
The following data are from a study of 19 children. The variables Height, Weight, and Age are measured for each child.

```
data Class;
   input Name $ Height Weight Age @@;
   datalines;
Alfred  69.0 112.5 14  Alice   56.5  84.0 13  Barbara 65.3  98.0 13
Carol   62.8 102.5 14  Henry   63.5 102.5 14  James   57.3  83.0 12
Jane    59.8  84.5 12  Janet   62.5 112.5 15  Jeffrey 62.5  84.0 13
John    59.0  99.5 12  Joyce   51.3  50.5 11  Judy    64.3  90.0 14
Louise  56.3  77.0 12  Mary    66.5 112.0 15  Philip  72.0 150.0 16
Robert  64.8 128.0 12  Ronald  67.0 133.0 15  Thomas  57.5  85.0 11
William 66.5 112.0 15
;
```

The following statements invoke the REG procedure to fit a simple linear regression model in which Weight is the response variable and Height is the independent variable, and select the diagnostic panel in Output 9.14.1.

```
ods html;
ods graphics on;
ods select DiagnosticsPanel;

proc reg data = Class;
   model Weight = Height;
run;
quit;

ods graphics off;
ods html close;
```

Output 9.14.1. Diagnostics Panel Annotated to Indicate Layout Structure



In the discussion that follows, the panel is modified so that it includes only the following components:

1. residual by predicted plot
4. residual Q-Q plot
6. Cook's D plot
7. residual histogram
9. summary statistics box

The panel to be produced is shown in Output 9.14.2. It displays components 1, 4, 6, and 7 in a 2×2 lattice, and it displays four of the summary statistics in component 9 in a box at the bottom.

The template that defines the original panel is "Ets.Reg.Graphics.DiagnosticPanel." The following listing is abbreviated to show the main structure of the template definition (see page 288 for details on how to display the complete template definition).

```
proc template;
define statgraph Stat.Reg.Graphics.DiagnosticsPanel;

   /* Dynamic variables */
   dynamic _TITLE _MODELLABEL _DEPLABEL _NOBS _NPARM _EDF _MSE
      _RSquare _AdjRSq;

   /* 3x3 LATTICE layout */
   layout lattice / columns = 3 rows = 3 ... ;

      sidebar / align=top;
         /* Statements for model label and graph title */
      endsidebar;

      /* 1. Residual By Predicted */
      layout overlay / ... ;
         lineparm slope = 0 yintercept = 0;
         scatterplot y = RESIDUAL x = PREDICTEDVALUE;
      endlayout;

      ...

      /* LAYOUT statements for components 2-8 */

      ...

      /* 9. Summary Statistics Box */
      layout overlay;
         layout gridded / ... ;
            entry "NObs";
            entry _NOBS / format=best6.;
               .
               .
               .
            entry "AdjRSq";
            entry _ADJRSQ / format=best6.;
         endlayout;
      endlayout;

   endlayout;   /* End of 3x3 LATTICE layout */
end;
run;
```

The overall display is defined by the LAYOUT LATTICE statement, which specifies a lattice of components, indicated by the solid grid annotated in Output 9.14.1. The COLUMNS=3 and ROWS=3 options in the LAYOUT LATTICE statement specify a 3×3 lattice, indicated by the dashed grid.

The model label and the graph title (top rectangle in Output 9.14.1) are specified inside the LATTICE layout with a SIDEBAR statement. The ALIGN=TOP option positions the sidebar at the top.

Each of the nine components of the lattice is defined by a LAYOUT statement. These statements define the components from left to right and top to bottom. Components 1 through 7 are defined with LAYOUT OVERLAY statements. Component 8 (RF plot) is defined with a LAYOUT LATTICE statement. The last LAYOUT OVERLAY statement defines a box with summary statistics for the fitted model.

The following abbreviated listing shows the basic structure of the template definition for a simplified panel that displays components 1, 4, 6, and 7 in a 2×2 lattice.[*] For the complete template definition, refer to the sample program odsgex14.sas in the SAS Sample Library for SAS/ETS software.

```
proc template;
define statgraph Stat.Reg.Graphics.DiagnosticsPanel;
   dynamic _TITLE _MODELLABEL _DEPLABEL _NOBS _NPARM _EDF _MSE
      _RSquare _AdjRSq;

   /* 2x2 LATTICE layout */
   /* Change COLUMNS= and ROWS= options */
   layout lattice / columns = 2 rows = 2 ... ;

      sidebar / align=top;
         /* Statements for model label and graph title */
      endsidebar;

      /* 1. Residual By Predicted */
      layout overlay / ... ;
         lineparm slope = 0 yintercept = 0;
         scatterplot y = RESIDUAL x = PREDICTEDVALUE;
      endlayout;

      /* 4. Q-Q Plot */
      layout overlay / ... ;
         lineparm slope      = eval(STDDEV(RESIDUAL))
                  yintercept = eval(...);
         scatterplot y = eval(...) x = eval(...);
      endlayout;

      /* Statements for components 6 and 7 (not listed) */

      /* Summary Statistics Box in a SIDEBAR */
      sidebar / align=bottom;
         layout gridded;
            layout lattice / rows=1 columns=4 ... ;
               .
               .
               .
            endlayout;
         endlayout;
      endsidebar;

   endlayout;   /* End of 2x2 LATTICE layout */
end;
run;
```

This template is a straightforward modification of the original template. The COLUMNS=2 and ROWS=2 options in the LAYOUT LATTICE statement request a 2×2 lattice. The LAYOUT statements for components 2, 3, 5, and 8 are deleted.

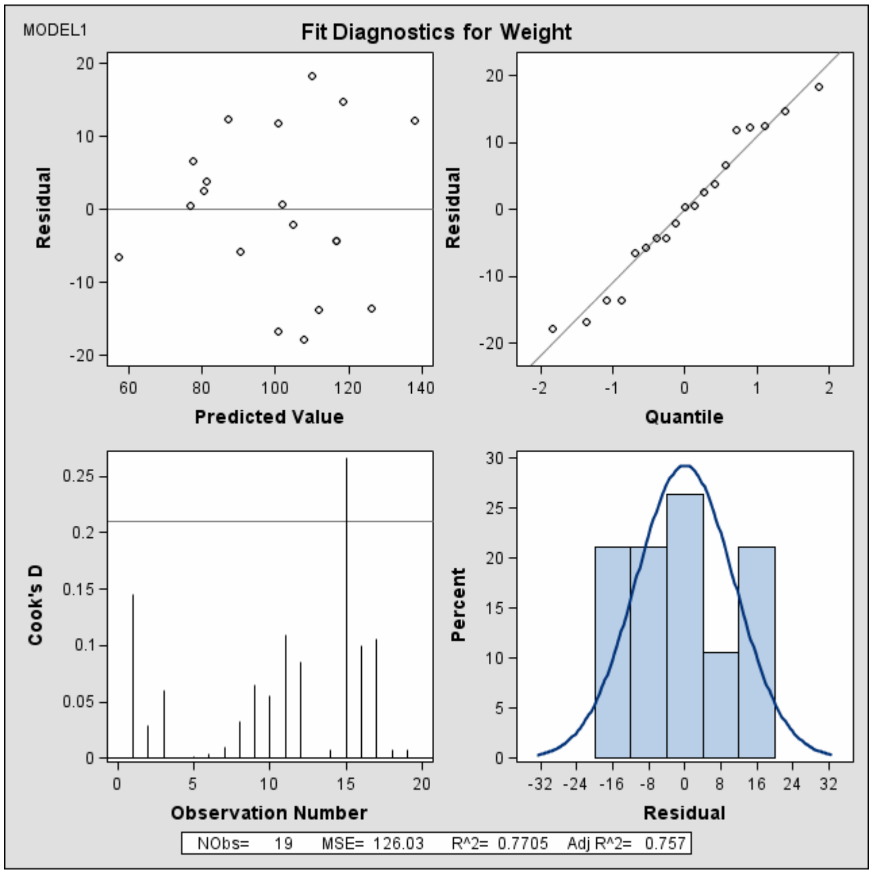
[*]See page 289 for details on how to edit the template definition.

A subset of the summary statistics are displayed at the bottom of the graph using a SIDEBAR statement with the ALIGN=BOTTOM option.

After submitting the preceding statements, which create the modified template and save it in Sasuser.Templat, you can run the following PROC REG statements to obtain the simplified panel, which is shown in Output 9.14.2.

```
ods html;
ods graphics on;

ods select DiagnosticsPanel;

proc reg data = Class;
   model Weight = Height;
run;
quit;

ods graphics off;
ods html close;
```

Output 9.14.2. Simplified Diagnostics Panel



```
NObs=    19    MSE= 126.03    R^2= 0.7705    Adj R^2= 0.757
```

A sample program named odsgr14.sas is available for this example in the SAS Sample Library for SAS/ETS software.

References

Houghton, A. N., Flannery, J., and Viola, M. V. (1980), “Malignant Melanoma in Connecticut and Denmark,” *International Journal of Cancer*, 25, 95–104.

Chapter 10
Nonlinear Optimization Methods

Chapter Contents

Chapter 10
Nonlinear Optimization Methods

Overview

Several SAS/ETS procedures (ENTROPY, MDC, QLIM, UCM, VARMAX) use the NonLinear Optimization (NLO) subsystem to perform nonlinear optimization. This chapter describes the options of the NLO system and some technical details of the available optimization methods. Note that not all options have been implemented for all procedures that use the NLO susbsystem. You should check each procedure chapter for more details on which options are available.

Options

The following table summarizes the options available in the NLO system.

Table 10.1. NLO options

Option	Description
Optimization Specifications	
TECHNIQUE=	minimization technique
UPDATE=	update technique
LINESEARCH=	line-search method
LSPRECISION=	line-search precision
HESCAL=	type of Hessian scaling
INHESSIAN=	start for approximated Hessian
RESTART=	iteration number for update restart
Termination Criteria Specifications	
MAXFUNC=	maximum number of function calls
MAXITER=	maximum number of iterations
MINITER=	minimum number of iterations
MAXTIME=	upper limit seconds of CPU time
ABSCONV=	absolute function convergence criterion
ABSFCONV=	absolute function convergence criterion
ABSGCONV=	absolute gradient convergence criterion
ABSXCONV=	absolute parameter convergence criterion
FCONV=	relative function convergence criterion
FCONV2=	relative function convergence criterion
GCONV=	relative gradient convergence criterion
XCONV=	relative parameter convergence criterion
FSIZE=	used in FCONV, GCONV criterion
XSIZE=	used in XCONV criterion
Step Length Options	
DAMPSTEP=	damped steps in line search
MAXSTEP=	maximum trust region radius

Table 10.1. (continued)

Option	Description
INSTEP=	initial trust region radius
Printed Output Options	
PALL	display (almost) all printed output
PHISTORY	display optimization history
PHISTPARMS	display parameter estimates in each iteration
PSHORT	reduce some default output
PSUMMARY	reduce most default output
NOPRINT	suppress all printed output
Remote Monitoring Options	
SOCKET=	specify the fileref for remote monitoring

These options are described in alphabetical order.

ABSCONV=r
ABSTOL=r
specifies an absolute function convergence criterion. For minimization, termination requires $f(\theta^{(k)}) \leq r$. The default value of r is the negative square root of the largest double precision value, which serves only as a protection against overflows.

ABSFCONV=$r[n]$
ABSFTOL=$r[n]$
specifies an absolute function convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 0$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=$r[n]$
ABSGTOL=$r[n]$
specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r = 1\text{E}{-5}$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSXCONV=$r[n]$
ABSXTOL=$r[n]$
specifies an absolute parameter convergence criterion. For all techniques except

NMSIMP, termination requires a small Euclidean distance between successive parameter vectors,

$$\| \theta^{(k)} - \theta^{(k-1)} \|_2 \leq r$$

For the NMSIMP technique, termination requires either a small length $\alpha^{(k)}$ of the vertices of a restart simplex,

$$\alpha^{(k)} \leq r$$

or a small simplex size,

$$\delta^{(k)} \leq r$$

where the simplex size $\delta^{(k)}$ is defined as the L1 distance from the simplex vertex $\xi^{(k)}$ with the smallest function value to the other n simplex points $\theta_l^{(k)} \neq \xi^{(k)}$:

$$\delta^{(k)} = \sum_{\theta_l \neq y} \| \theta_l^{(k)} - \xi^{(k)} \|_1$$

The default is $r = 1\text{E} - 8$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

DAMPSTEP[=r]

specifies that the initial step length value $\alpha^{(0)}$ for each line search (used by the QUANEW, HYQUAN, CONGRA, or NEWRAP technique) cannot be larger than r times the step length value used in the former iteration. If the DAMPSTEP option is specified but r is not specified, the default is $r = 2$. The DAMPSTEP=r option can prevent the line-search algorithm from repeatedly stepping into regions where some objective functions are difficult to compute or where they could lead to floating point overflows during the computation of objective functions and their derivatives. The DAMPSTEP=r option can save time-costly function calls during the line searches of objective functions that result in very small steps.

FCONV=$r[n]$
FTOL=$r[n]$

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{\max(|f(\theta^{(k-1)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value may depend on the procedure. In most cases, you can use the PALL option to find it.

FCONV2=$r[n]$
FTOL2=$r[n]$

specifies another function convergence criterion. For all techniques except NMSIMP, termination requires a small predicted reduction

$$df^{(k)} \approx f(\theta^{(k)}) - f(\theta^{(k)} + s^{(k)})$$

of the objective function. The predicted reduction

$$\begin{aligned} df^{(k)} &= -g^{(k)T}s^{(k)} - \frac{1}{2}s^{(k)T}H^{(k)}s^{(k)} \\ &= -\frac{1}{2}s^{(k)T}g^{(k)} \\ &\leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step.

$$s^{(k)} = -[H^{(k)}]^{-1}g^{(k)}$$

For the NMSIMP technique, termination requires a small standard deviation of the function values of the $n+1$ simplex vertices $\theta_l^{(k)}$, $l = 0, \ldots, n$,

$$\sqrt{\frac{1}{n+1}\sum_l \left[f(\theta_l^{(k)}) - \overline{f}(\theta^{(k)})\right]^2} \leq r$$

where $\overline{f}(\theta^{(k)}) = \frac{1}{n+1}\sum_l f(\theta_l^{(k)})$. If there are n_{act} boundary constraints active at $\theta^{(k)}$, the mean and standard deviation are computed only for the $n+1-n_{act}$ unconstrained vertices. The default value is $r = 1\text{E}-6$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FSIZE=r

specifies the FSIZE parameter of the relative function and relative gradient termination criteria. The default value is $r = 0$. For more details, see the FCONV= and GCONV= options.

GCONV=$r[n]$
GTOL=$r[n]$

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction is small,

$$\frac{g(\theta^{(k)})^T[H^{(k)}]^{-1}g(\theta^{(k)})}{\max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\parallel g(\theta^{(k)}) \parallel_2^2 \quad \parallel s(\theta^{(k)}) \parallel_2}{\parallel g(\theta^{(k)}) - g(\theta^{(k-1)}) \parallel_2 \max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r = 1E-8$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

HESCAL=$0|1|2|3$
HS=$0|1|2|3$

specifies the scaling version of the Hessian matrix used in NRRIDG, TRUREG, NEWRAP, or DBLDOG optimization. If HS is not equal to 0, the first iteration and each restart iteration sets the diagonal scaling matrix $D^{(0)} = \mathrm{diag}(d_i^{(0)})$:

$$d_i^{(0)} = \sqrt{\max(|H_{i,i}^{(0)}|, \epsilon)}$$

where $H_{i,i}^{(0)}$ are the diagonal elements of the Hessian. In every other iteration, the diagonal scaling matrix $D^{(0)} = \mathrm{diag}(d_i^{(0)})$ is updated depending on the HS option:

HS=0	specifies that no scaling is done.
HS=1	specifies the Moré (1978) scaling update: $$d_i^{(k+1)} = \max\left[d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$
HS=2	specifies the Dennis, Gay, & Welsch (1981) scaling update: $$d_i^{(k+1)} = \max\left[0.6 * d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$
HS=3	specifies that d_i is reset in each iteration: $$d_i^{(k+1)} = \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}$$

In each scaling update, ϵ is the relative machine precision. The default value is HS=0. Scaling of the Hessian can be time consuming in the case where general linear constraints are active.

INHESSIAN[= r**]**
INHESS[= r**]**

specifies how the initial estimate of the approximate Hessian is defined for the quasi-Newton techniques QUANEW and DBLDOG. There are two alternatives:

- If you do not use the r specification, the initial estimate of the approximate Hessian is set to the Hessian at $\theta^{(0)}$.
- If you do use the r specification, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix rI.

By default, if you do not specify the option INHESSIAN=r, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix rI, where the scalar r is computed from the magnitude of the initial gradient.

INSTEP=r

reduces the length of the first trial step during the line search of the first iterations. For highly nonlinear objective functions, such as the EXP function, the default initial radius of the trust-region algorithm TRUREG or DBLDOG or the default step length of the line-search algorithms can result in arithmetic overflows. If this occurs, you should specify decreasing values of $0 < r < 1$ such as INSTEP=1E $-$ 1, INSTEP=1E $-$ 2, INSTEP=1E $-$ 4, and so on, until the iteration starts successfully.

- For trust-region algorithms (TRUREG, DBLDOG), the INSTEP= option specifies a factor $r > 0$ for the initial radius $\Delta^{(0)}$ of the trust region. The default initial trust-region radius is the length of the scaled gradient. This step corresponds to the default radius factor of $r = 1$.
- For line-search algorithms (NEWRAP, CONGRA, QUANEW), the INSTEP= option specifies an upper bound for the initial step length for the line search during the first five iterations. The default initial step length is $r = 1$.
- For the Nelder-Mead simplex algorithm, using TECH=NMSIMP, the INSTEP=r option defines the size of the start simplex.

LINESEARCH=i
LIS=i

specifies the line-search method for the CONGRA, QUANEW, and NEWRAP optimization techniques. Refer to Fletcher (1987) for an introduction to line-search techniques. The value of i can be $1, \ldots, 8$. For CONGRA, QUANEW and NEWRAP, the default value is $i = 2$.

LIS=1	specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is similar to one used by the Harwell subroutine library.
LIS=2	specifies a line-search method that needs more function than gradient calls for quadratic and cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.
LIS=3	specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.
LIS=4	specifies a line-search method that needs the same number of function and gradient calls for stepwise extrapolation and cubic interpolation.
LIS=5	specifies a line-search method that is a modified version of LIS=4.
LIS=6	specifies golden section line search (Polak 1971), which uses only function values for linear approximation.
LIS=7	specifies bisection line search (Polak 1971), which uses only function values for linear approximation.
LIS=8	specifies the Armijo line-search technique (Polak 1971), which uses only function values for linear approximation.

LSPRECISION=r
LSP=r

specifies the degree of accuracy that should be obtained by the line-search algorithms LIS=2 and LIS=3. Usually an imprecise line search is inexpensive and successful. For more difficult optimization problems, a more precise and expensive line search may be necessary (Fletcher 1987). The second line-search method (which is the default for the NEWRAP, QUANEW, and CONGRA techniques) and the third line-search method approach exact line search for small LSPRECISION= values. If you have numerical problems, you should try to decrease the LSPRECISION= value to obtain a more precise line search. The default values are shown in the following table.

TECH=	**UPDATE=**	**LSP default**
QUANEW	DBFGS, BFGS	r = 0.4
QUANEW	DDFP, DFP	r = 0.06
CONGRA	all	r = 0.1
NEWRAP	no update	r = 0.9

For more details, refer to Fletcher (1987).

MAXFUNC=i
MAXFU=i

specifies the maximum number i of function calls in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 125
- QUANEW, DBLDOG: 500
- CONGRA: 1000
- NMSIMP: 3000

Note that the optimization can terminate only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that is specified by the MAXFUNC= option.

MAXITER=i
MAXIT=i

specifies the maximum number i of iterations in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 50
- QUANEW, DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1000

These default values are also valid when i is specified as a missing value.

MAXSTEP=$r[n]$

specifies an upper bound for the step length of the line-search algorithms during the first n iterations. By default, r is the largest double precision value and n is the largest integer available. Setting this option can improve the speed of convergence for the CONGRA, QUANEW, and NEWRAP techniques.

MAXTIME=r

specifies an upper limit of r seconds of CPU time for the optimization process. The default value is the largest floating point double representation of your computer. Note that the time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be much longer than that specified by the MAXTIME= option. The actual running time includes the rest of the time needed to finish the iteration and the time needed to generate the output of the results.

MINITER=i
MINIT=i

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NOPRINT

suppresses the output. (See proc documentation for availability of this option.)

PALL

displays all optional output for optimization. (See proc documentation for availability of this option.)

PHISTORY

displays the optimization history. (See proc documentation for availability of this option.)

PHISTPARMS

display parameter estimates in each iteration. (See proc documentation for availability of this option.)

PINIT

displays the initial values and derivatives (if available). (See proc documentation for availability of this option.)

PSHORT

restricts the amount of default output. (See proc documentation for availability of this option.)

PSUMMARY

restricts the amount of default displayed output to a short form of iteration history and notes, warnings, and errors. (See proc documentation for availability of this option.)

RESTART=$i > 0$
REST=$i > 0$

specifies that the QUANEW or CONGRA algorithm is restarted with a steepest descent/ascent search direction after, at most, i iterations. Default values are

- CONGRA: UPDATE=PB: restart is performed automatically, i is not used.

- CONGRA: UPDATE≠PB: $i = \min(10n, 80)$, where n is the number of parameters.
- QUANEW: i is the largest integer available.

SOCKET=*fileref*
Specifies the fileref that contains the information needed for remote monitoring. See the section "Remote Monitoring" on page 353 for more details.

TECHNIQUE=*value*
TECH=*value*
specifies the optimization technique. Valid values are

- CONGRA
  performs a conjugate-gradient optimization, which can be more precisely specified with the UPDATE= option and modified with the LINESEARCH= option. When you specify this option, UPDATE=PB by default.
- DBLDOG
  performs a version of double dogleg optimization, which can be more precisely specified with the UPDATE= option. When you specify this option, UPDATE=DBFGS by default.
- NMSIMP
  performs a Nelder-Mead simplex optimization.
- NONE
  does not perform any optimization. This option can be used
  - to perform a grid search without optimization
  - to compute estimates and predictions that cannot be obtained efficiently with any of the optimization techniques
- NEWRAP
  performs a Newton-Raphson optimization combining a line-search algorithm with ridging. The line-search algorithm LIS=2 is the default method.
- NRRIDG
  performs a Newton-Raphson optimization with ridging.
- QUANEW
  performs a quasi-Newton optimization, which can be defined more precisely with the UPDATE= option and modified with the LINESEARCH= option. This is the default estimation method.
- TRUREG
  performs a trust region optimization.

UPDATE=*method*
UPD=*method*
specifies the update method for the quasi-Newton, double dogleg, or conjugate-gradient optimization technique. Not every update method can be used with each optimizer.

Valid methods are

- BFGS
  performs the original Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the inverse Hessian matrix.
- DBFGS
  performs the dual BFGS update of the Cholesky factor of the Hessian matrix. This is the default update method.
- DDFP
  performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- DFP
  performs the original DFP update of the inverse Hessian matrix.
- PB
  performs the automatic restart update method of Powell (1977) and Beale (1972).
- FR
  performs the Fletcher-Reeves update (Fletcher 1987).
- PR
  performs the Polak-Ribiere update (Fletcher 1987).
- CD
  performs a conjugate-descent update of Fletcher (1987).

XCONV=$r[n]$
XTOL=$r[n]$

specifies the relative parameter convergence criterion. For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations.

$$\frac{\max_j |\theta_j^{(k)} - \theta_j^{(k-1)}|}{\max(|\theta_j^{(k)}|, |\theta_j^{(k-1)}|, \text{XSIZE})} \leq r$$

For the NMSIMP technique, the same formula is used, but $\theta_j^{(k)}$ is defined as the vertex with the lowest function value and $\theta_j^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 1\text{E} - 8$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

XSIZE=$r > 0$

specifies the XSIZE parameter of the relative parameter termination criterion. The default value is $r = 0$. For more detail, see the XCONV= option.

Details of Optimization Algorithms

Overview

There are several optimization techniques available. You can choose a particular optimizer with the TECH=$name$ option in the PROC statement or NLOPTIONS statement.

Algorithm	TECH=
trust region Method	TRUREG
Newton-Raphson method with line search	NEWRAP
Newton-Raphson method with ridging	NRRIDG
quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)	QUANEW
double-dogleg method (DBFGS, DDFP)	DBLDOG
conjugate gradient methods (PB, FR, PR, CD)	CONGRA
Nelder-Mead simplex method	NMSIMP

No algorithm for optimizing general nonlinear functions exists that always finds the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, NLO provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in NLO can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem.

All optimization techniques in NLO use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters. Since the techniques are iterative, they require the repeated computation of

- the function value (optimization criterion)
- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)

However, since each of the optimizers requires different derivatives, some computational efficiencies can be gained. The following table shows, for each optimization technique, which derivatives are required (FOD: first-order derivatives; SOD: second-order derivatives).

Algorithm	FOD	SOD
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

Each optimization method employs one or more convergence criteria that determine when it has converged. The various termination criteria are listed and described in the previous section. An algorithm is considered to have converged when any one of the convergence criterion is satisfied. For example, under the default settings, the QUANEW algorithm will converge if $ABSGCONV < 1\text{E}-5$, $FCONV < 10^{-FDIGITS}$, or $GCONV < 1\text{E}-8$.

Choosing an Optimization Algorithm

The factors that go into choosing a particular optimization technique for a particular problem are complex and may involve trial and error.

For many optimization problems, computing the gradient takes more computer time than computing the function value, and computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix, and as a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can more easily terminate at stationary points rather than at global optima.

A few general remarks about the various optimization techniques follow.

- The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems where the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $n(n+1)/2$ double words; TRUREG and NEWRAP require two such matrices.
- The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems where the objective function and the gradient are much faster to evaluate than the Hessian. The QUANEW and DBLDOG algorithms, in general, require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP (essentially one matrix with $n(n+1)/2$ double words). QUANEW is the default optimization method.

- The first-derivative method CONGRA is best for large problems where the objective function and the gradient can be computed much faster than the Hessian and where too much memory is required to store the (approximate) Hessian. The CONGRA algorithm, in general, requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Since CONGRA requires only a factor of n double-word memory, many large applications can be solved only by CONGRA.
- The no-derivative method NMSIMP is best for small problems where derivatives are not continuous or are very difficult to compute.

Algorithm Descriptions

Some details about the optimization techniques are as follows.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function $f(\theta)$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region with radius Δ that constrains the step size corresponding to the quality of the quadratic approximation. The trust region method is implemented using Dennis, Gay, and Welsch (1981), Gay (1983), and Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the (dual) quasi-Newton or conjugate gradient algorithms may be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region. If second-order derivatives are computed efficiently and precisely, the NEWRAP method may perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation (LIS=2).

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the (dual) quasi-Newton or conjugate gradient algorithms may be more efficient.

Since the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than that of the NEWRAP technique, which works with Cholesky decomposition. Usually, however, NRRIDG requires fewer iterations than NEWRAP.

Quasi-Newton Optimization (QUANEW)

The (dual) quasi-Newton method uses the gradient $g(\theta_{(k)})$, and it does not need to compute second-order derivatives since they are approximated. It works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian; but, in general, it requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. QUANEW is the default optimization algorithm because it provides an appropriate balance between the speed and stability required for most nonlinear mixed model applications.

The QUANEW technique is one of the following, depending upon the value of the UPDATE= option.

- the original quasi-Newton algorithm, which updates an approximation of the inverse Hessian
- the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian (default)

You can specify four update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- BFGS performs the original BFGS update of the inverse Hessian matrix.
- DFP performs the original DFP update of the inverse Hessian matrix.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α satisfying the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted with an identity matrix, resulting in the steepest descent or ascent search direction. You can specify line-search algorithms other than the default with the LIS= option.

The QUANEW algorithm performs its own line-search technique. All options and parameters (except the INSTEP= option) controlling the line search in the other algorithms do not apply here. In several applications, large steps in the first iterations are troublesome. You can use the INSTEP= option to impose an upper bound for the step size α during the first five iterations. You can also use the INHESSIAN[=r] option to specify a different starting approximation for the Hessian. If you specify only the INHESSIAN option, the Cholesky factor of a (possibly ridged) finite difference approximation of the Hessian is used to initialize the quasi-Newton update process. The values of the LCSINGULAR=, LCEPSILON=, and LCDEACT= options, which control the processing of linear and boundary constraints, are valid only for the quadratic programming subroutine used in each iteration of the QUANEW algorithm.

Double Dogleg Optimization (DBLDOG)

The double dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double dogleg algorithm computes the step $s^{(k)}$ as the linear combination of the steepest descent or ascent search direction $s_1^{(k)}$ and a quasi-Newton search direction $s_2^{(k)}$.

$$s^{(k)} = \alpha_1 s_1^{(k)} + \alpha_2 s_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius; refer to Fletcher (1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search. You can specify two update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell update of the Cholesky factor of the Hessian matrix.

The double dogleg optimization technique works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian. The implementation is based on Dennis and Mei (1979) and Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, or NRRIDG technique, which requires second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(n)$ memory for unconstrained optimization. In general, many iterations are required to obtain a precise solution, but each of the CONGRA iterations is computationally cheap. You can specify four different update formulas for generating the conjugate directions by using the UPDATE= option:

- PB performs the automatic restart update method of Powell (1977) and Beale (1972). This is the default.
- FR performs the Fletcher-Reeves update (Fletcher 1987).
- PR performs the Polak-Ribiere update (Fletcher 1987).
- CD performs a conjugate-descent update of Fletcher (1987).

The default, UPDATE=PB, behaved best in most test examples. You are advised to avoid the option UPDATE=CD, which behaved worst in most test examples.

The CONGRA subroutine should be used for optimization problems with large n. For the unconstrained or boundary constrained case, CONGRA requires only $O(n)$ bytes of working memory, whereas all other optimization methods require order $O(n^2)$ bytes of working memory. During n successive iterations, uninterrupted by restarts or changes in the working set, the conjugate gradient algorithm computes a cycle of n conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α satisfying the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size. Other line-search algorithms can be specified with the LIS= option.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it may be unable to generate precise results for $n \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex adapting to the nonlinearities of the objective function, which contributes to an increased speed of convergence. It uses a special termination criteria.

Remote Monitoring

The SAS/EmMonitor is an application for Windows that enables you to monitor and stop from your PC a CPU-intensive application performed by the NLO subsystem running on a remote server.

On the server side, a FILENAME statement assigns a fileref to a SOCKET-type device that defines the ip address of the client and the port number for listening. The fileref is then specified in the SOCKET= option to control the EmMonitor. The following statements show an example of server-side code for PROC ENTROPY.

```
data one;
 do t = 1 to 10;
    x1 = 5 * ranuni(456);
    x2 = 10 * ranuni( 456);
    x3 = 2 * rannor(1456);
    e1 = rannor(1456);
    e2 = rannor(4560);
    tmp1 = 0.5 * e1 - 0.1 * e2;
    tmp2 = -0.1 * e1 - 0.3 * e2;
    y1 = 7 + 8.5*x1 + 2*x2 + tmp1;
    y2 = -3 + -2*x1 + x2 + 3*x3 + tmp2;
    output;
   end;
run;

filename sock socket 'your.pc.address.com:6943';

proc entropy data=one tech=tr gmenm gconv=2.e-5 socket=sock;
   model y1 = x1 x2 x3;
run;
```

On the client side, the EmMonitor application is started with the following syntax:

EmMonitor *options*

The options are:

-p port_number	define the port number
-t title	define the title of the EmMonitor window
-k	keep the monitor alive when the iteration is completed

The default port number is 6943.

The server does not need to be running when you start the EmMonitor, and you can start or dismiss it at any time during the iteration process. You only need to remember the port number.

If you do not start the PC client, or you close it prematurely, it will not have any effect on the server side. In other words, the iteration process will continue until one of the criteria for termination is met.

Figure 10.1 through Figure 10.4 show screenshots of the application on the client side.

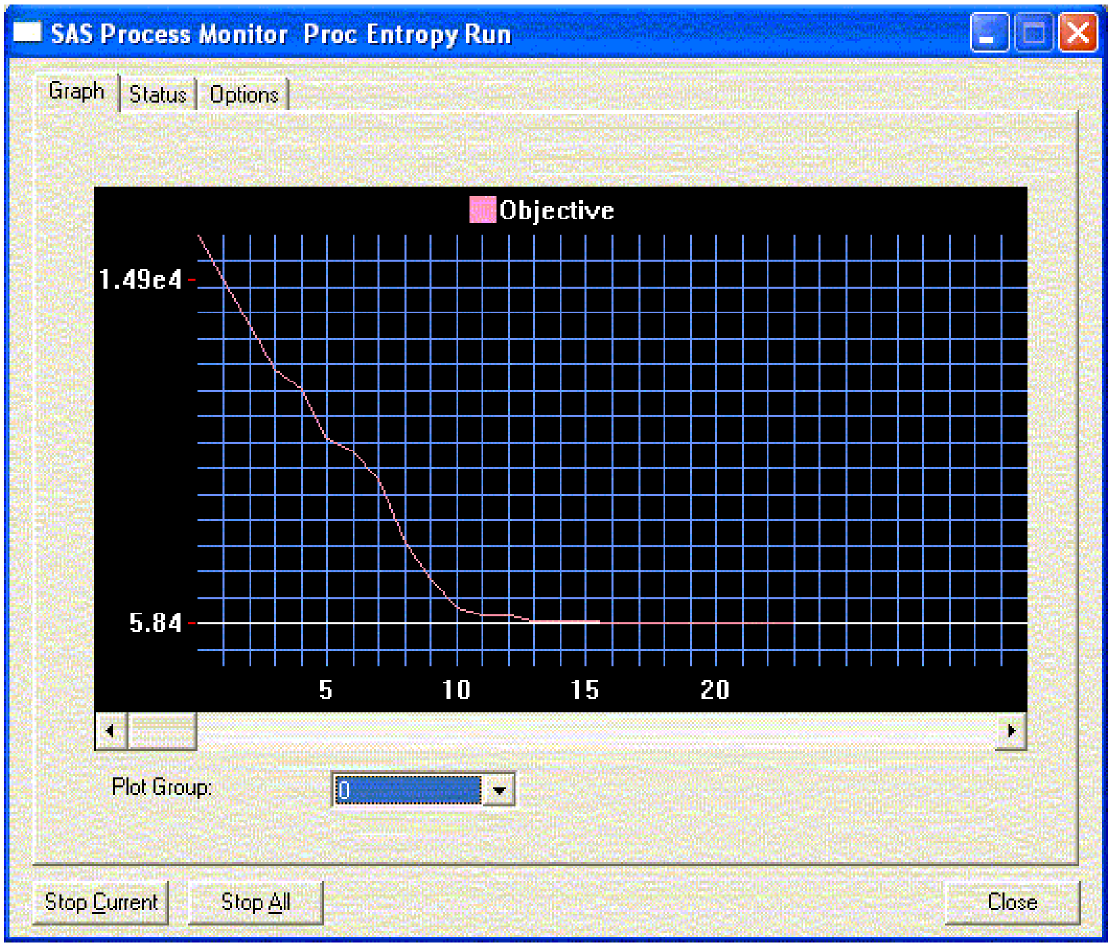


Figure 10.1. Graph Tab Group 0

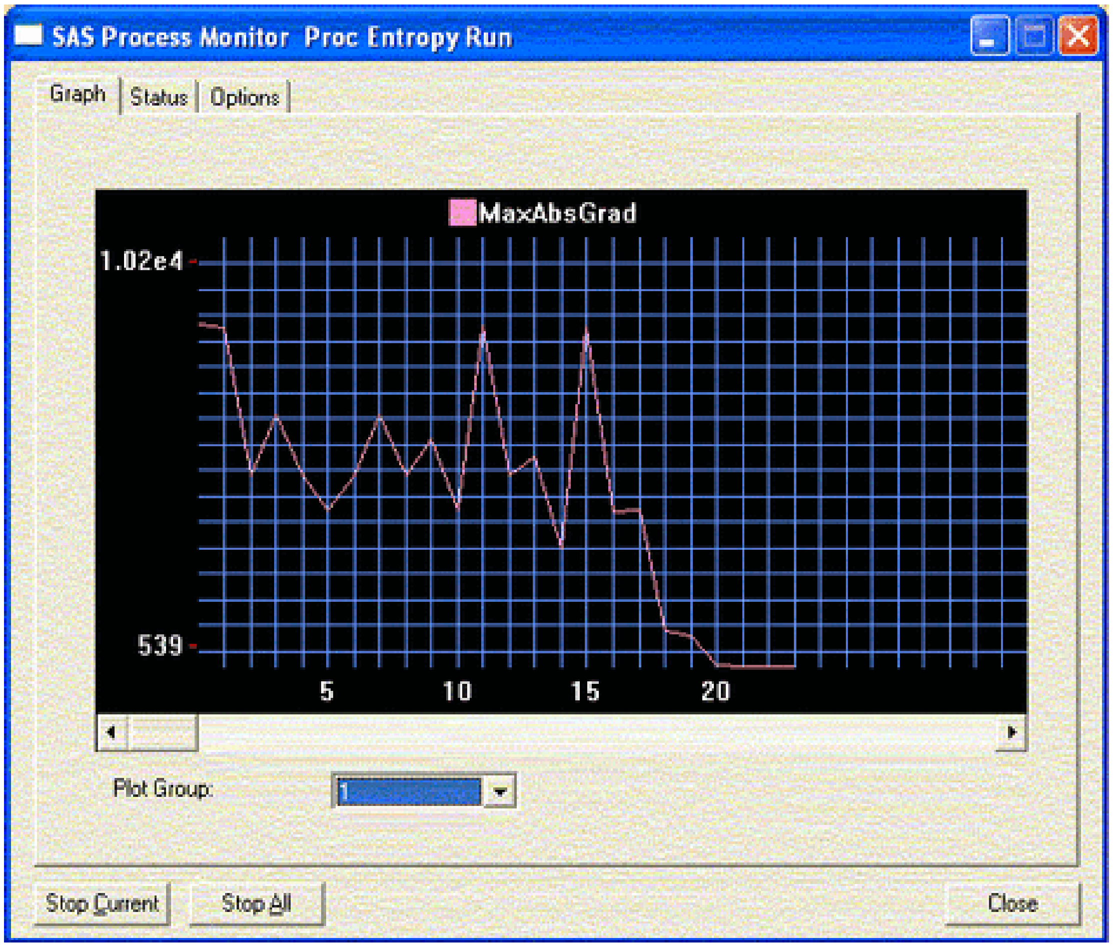
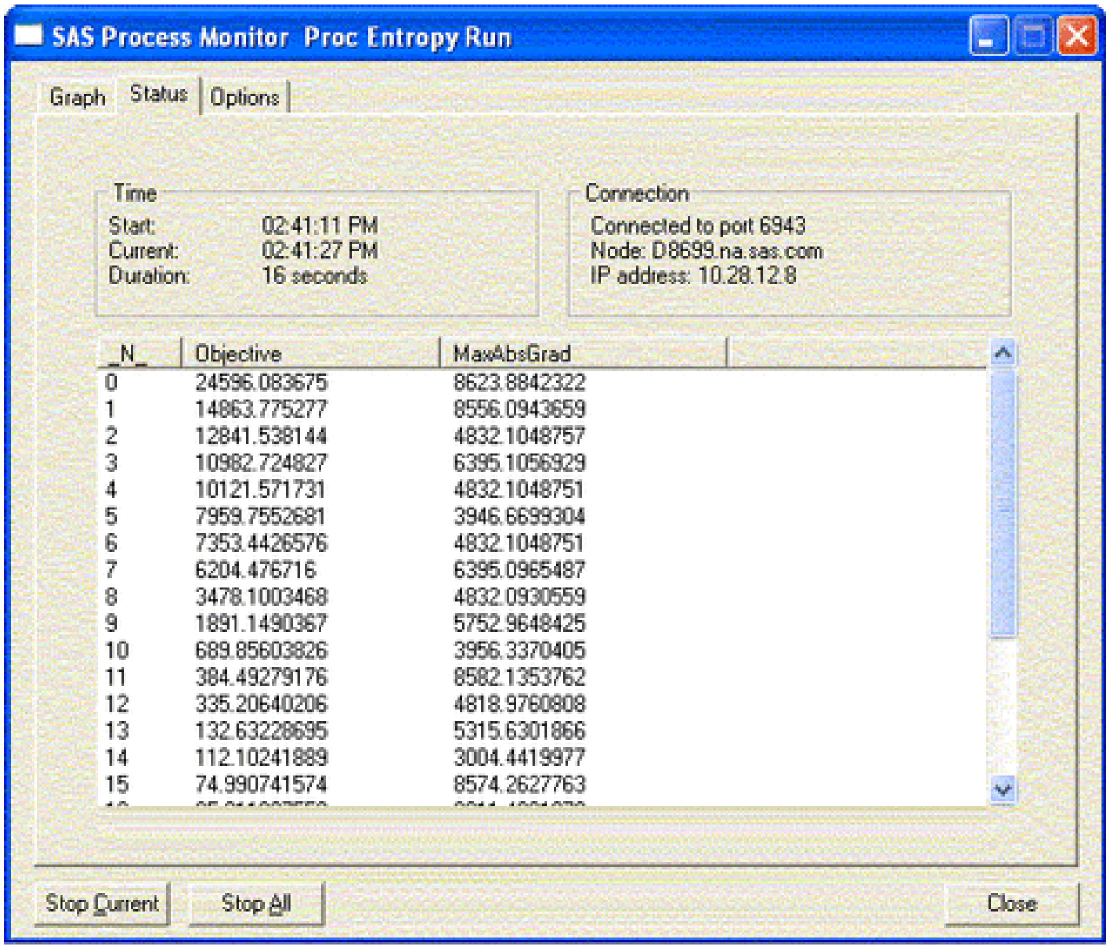


Figure 10.2. Graph Tab Group 1



Time

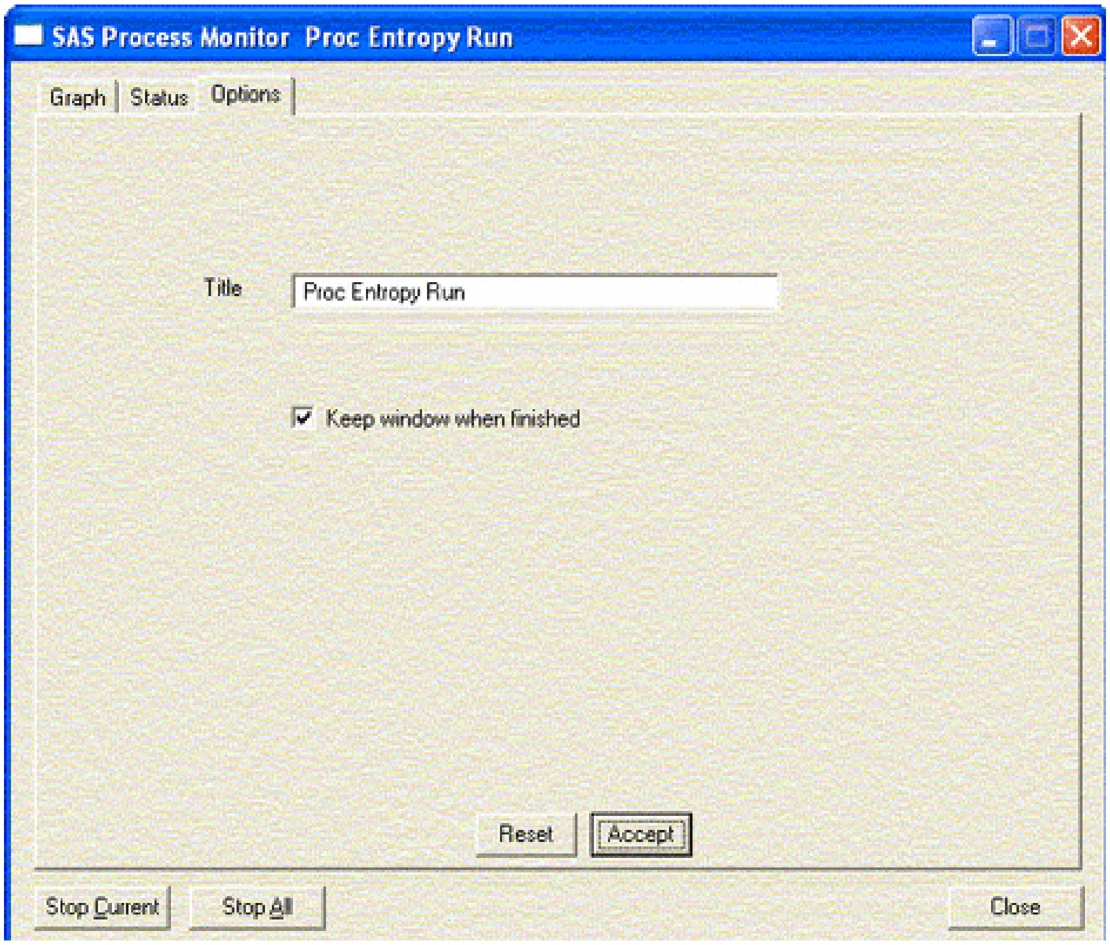
Start:	02:41:11 PM
Current:	02:41:27 PM
Duration:	16 seconds

Connection

Connected to port 6943
Node: D8699.na.sas.com
IP address: 10.28.12.8

_N_	Objective	MaxAbsGrad
0	24596.083675	8623.8842322
1	14863.775277	8556.0943659
2	12841.538144	4832.1048757
3	10982.724827	6395.1056929
4	10121.571731	4832.1048751
5	7959.7552681	3946.6699304
6	7353.4426576	4832.1048751
7	6204.476716	6395.0965487
8	3478.1003468	4832.0930559
9	1891.1490367	5752.9648425
10	689.85603826	3956.3370405
11	384.49279176	8582.1353762
12	335.20640206	4818.9760808
13	132.63228695	5315.6301866
14	112.10241889	3004.4419977
15	74.990741574	8574.2627763

Figure 10.3. Status Tab



Title: Proc Entropy Run

☑ Keep window when finished

Figure 10.4. Options Tab

ODS Table Names

The NLO subsystem assigns a name to each table it creates. You can use these names when using the Output Delivery System (ODS) to select tables and create output data sets. Not all tables are created by all SAS/ETS procedures that use the NLO subsystem. You should check the procedure chapter for more details. The names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 10.2. ODS Tables Produced by the NLO Subsystem

ODS Table Name	Description
ConvergenceStatus	Convergence status
InputOptions	Input options
IterHist	Iteration history
IterStart	Iteration start
IterStop	Iteration stop
Lagrange	Lagrange multipliers at the solution
LinCon	Linear constraints
LinConDel	Deleted linear constraints
LinConSol	Linear constraints at the solution
ParameterEstimatesResults	Estimates at the results
ParameterEstimatesStart	Estimates at the start of the iterations
ProblemDescription	Problem description
ProjGrad	Projected gradients

References

Beale, E.M.L. (1972), "A Derivation of Conjugate Gradients," in *Numerical Methods for Nonlinear Optimization*, ed. F.A. Lootsma, London: Academic Press.

Dennis, J.E., Gay, D.M., and Welsch, R.E. (1981), "An Adaptive Nonlinear Least-Squares Algorithm," *ACM Transactions on Mathematical Software*, 7, 348–368.

Dennis, J.E. and Mei, H.H.W. (1979), "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values," *J. Optim. Theory Appl.*, 28, 453–482.

Dennis, J.E. and Schnabel, R.B. (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood, NJ: Prentice-Hall.

Fletcher, R. (1987), *Practical Methods of Optimization*, Second Edition, Chichester: John Wiley & Sons, Inc.

Gay, D.M. (1983), "Subroutines for Unconstrained Minimization," *ACM Transactions on Mathematical Software*, 9, 503–524.

Moré, J.J. (1978), “The Levenberg-Marquardt Algorithm: Implementation and Theory,” in *Lecture Notes in Mathematics 630*, ed. G.A. Watson, Berlin-Heidelberg-New York: Springer Verlag.

Moré, J.J. and Sorensen, D.C. (1983), “Computing a Trust-region Step,” *SIAM Journal on Scientific and Statistical Computing*, 4, 553–572.

Polak, E. (1971), *Computational Methods in Optimization*, New York: Academic Press.

Powell, J.M.D. (1977), “Restart Procedures for the Conjugate Gradient Method,” *Math. Prog.*, 12, 241–254.

Part 2

Procedure Reference

Contents

Procedure Reference

Chapter 11
The ARIMA Procedure

Chapter Contents

Chapter 11 The ARIMA Procedure

Overview

The ARIMA procedure analyzes and forecasts equally spaced univariate time series data, transfer function data, and intervention data using the **A**uto**R**egressive **I**ntegrated **M**oving-**A**verage (ARIMA) or autoregressive moving-average (ARMA) model. An ARIMA model predicts a value in a response time series as a linear combination of its own past values, past errors (also called shocks or innovations), and current and past values of other time series.

The ARIMA approach was first popularized by Box and Jenkins, and ARIMA models are often referred to as Box-Jenkins models. The general transfer function model employed by the ARIMA procedure was discussed by Box and Tiao (1975). When an ARIMA model includes other time series as input variables, the model is sometimes referred to as an ARIMAX model. Pankratz (1991) refers to the ARIMAX model as *dynamic regression*.

The ARIMA procedure provides a comprehensive set of tools for univariate time series model identification, parameter estimation, and forecasting, and it offers great flexibility in the kinds of ARIMA or ARIMAX models that can be analyzed. The ARIMA procedure supports seasonal, subset, and factored ARIMA models; intervention or interrupted time series models; multiple regression analysis with ARMA errors; and rational transfer function models of any complexity.

Experimental graphics are now available with the ARIMA procedure. For more information, see the "ODS Graphics" section on page 443.

The design of PROC ARIMA closely follows the Box-Jenkins strategy for time series modeling with features for the identification, estimation and diagnostic checking, and forecasting steps of the Box-Jenkins method.

Before using PROC ARIMA, you should be familiar with Box-Jenkins methods, and you should exercise care and judgment when using the ARIMA procedure. The ARIMA class of time series models is complex and powerful, and some degree of expertise is needed to use them correctly.

If you are unfamiliar with the principles of ARIMA modeling, refer to textbooks on time series analysis. Also refer to *SAS/ETS Software: Applications Guide 1, Version 6, First Edition*. You might consider attending the SAS Training Course "Introduction to Time Series Forecasting Using SAS/ETS Software." This course provides in-depth training on ARIMA modeling using PROC ARIMA, as well as training on the use of other forecasting tools available in SAS/ETS software.

Getting Started

This section outlines the use of the ARIMA procedure and gives a cursory description of the ARIMA modeling process for readers less familiar with these methods.

The Three Stages of ARIMA Modeling

The analysis performed by PROC ARIMA is divided into three stages, corresponding to the stages described by Box and Jenkins (1976).

1. In the *identification* stage, you use the IDENTIFY statement to specify the response series and identify candidate ARIMA models for it. The IDENTIFY statement reads time series that are to be used in later statements, possibly differencing them, and computes autocorrelations, inverse autocorrelations, partial autocorrelations, and cross correlations. Stationarity tests can be performed to determine if differencing is necessary. The analysis of the IDENTIFY statement output usually suggests one or more ARIMA models that could be fit. Options enable you to test for stationarity and tentative ARMA order identification.
2. In the *estimation and diagnostic checking* stage, you use the ESTIMATE statement to specify the ARIMA model to fit to the variable specified in the previous IDENTIFY statement, and to estimate the parameters of that model. The ESTIMATE statement also produces diagnostic statistics to help you judge the adequacy of the model.

   Significance tests for parameter estimates indicate whether some terms in the model may be unnecessary. Goodness-of-fit statistics aid in comparing this model to others. Tests for white noise residuals indicate whether the residual series contains additional information that might be utilized by a more complex model. The OUTLIER statement provides another useful tool to check whether the currently estimated model accounts for all the variation in the series. If the diagnostic tests indicate problems with the model, you try another model, then repeat the estimation and diagnostic checking stage.
3. In the *forecasting* stage you use the FORECAST statement to forecast future values of the time series and to generate confidence intervals for these forecasts from the ARIMA model produced by the preceding ESTIMATE statement.

These three steps are explained further and illustrated through an extended example in the following sections.

Identification Stage

Suppose you have a variable called SALES that you want to forecast. The following example illustrates ARIMA modeling and forecasting using a simulated data set TEST containing a time series SALES generated by an ARIMA(1,1,1) model. The output produced by this example is explained in the following sections. The simulated SALES series is shown in Figure 11.1.

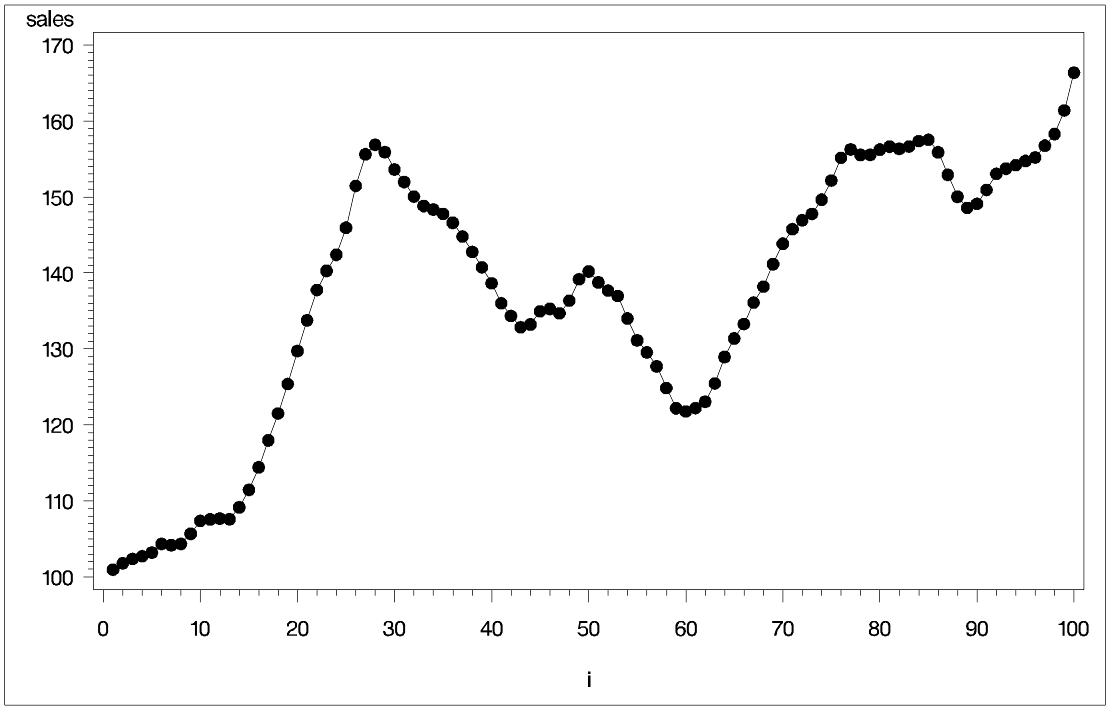


Figure 11.1. Simulated ARIMA(1, 1, 1) Series SALES

Using the IDENTIFY Statement

You first specify the input data set in the PROC ARIMA statement. Then, you use an IDENTIFY statement to read in the SALES series and plot its autocorrelation function. You do this using the following statements:

```
proc arima data=test;
   identify var=sales nlag=8;
   run;
```

Descriptive Statistics

The IDENTIFY statement first prints descriptive statistics for the SALES series. This part of the IDENTIFY statement output is shown in Figure 11.2.

```
                              The ARIMA Procedure

                            Name of Variable = sales

                       Mean of Working Series    137.3662
                       Standard Deviation        17.36385
                       Number of Observations         100
```

Figure 11.2. IDENTIFY Statement Descriptive Statistics Output

Autocorrelation Function Plots

The IDENTIFY statement next prints three plots of the correlations of the series with its past values at different lags. These are the

- sample autocorrelation function plot
- sample partial autocorrelation function plot
- sample inverse autocorrelation function plot

The sample autocorrelation function plot output of the IDENTIFY statement is shown in Figure 11.3.

```
                              The ARIMA Procedure

                                Autocorrelations

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0      301.503       1.00000   |                    |********************|
  1      288.454       0.95672   |                .   |******************* |
  2      273.437       0.90691   |             .      |******************  |
  3      256.787       0.85169   |            .       |*****************   |
  4      238.518       0.79110   |          .         |****************    |
  5      219.033       0.72647   |         .          |***************     |
  6      198.617       0.65876   |         .          |*************       |
  7      177.150       0.58755   |        .           |************        |
  8      154.914       0.51381   |       .            |**********  .       |

                         "." marks two standard errors
```

Figure 11.3. IDENTIFY Statement Autocorrelations Plot

The autocorrelation plot shows how values of the series are correlated with past values of the series. For example, the value 0.95672 in the “Correlation” column for the Lag 1 row of the plot means that the correlation between SALES and the SALES value for the previous period is .95672. The rows of asterisks show the correlation values graphically.

These plots are called autocorrelation functions because they show the degree of correlation with past values of the series as a function of the number of periods in the past (that is, the lag) at which the correlation is computed.

The NLAG= option controls the number of lags for which autocorrelations are shown. By default, the autocorrelation functions are plotted to lag 24; in this example the NLAG=8 option is used, so only the first 8 lags are shown.

Most books on time series analysis explain how to interpret autocorrelation plots and partial autocorrelation plots. See the section "The Inverse Autocorrelation Function" on page 409 for a discussion of inverse autocorrelation plots.

By examining these plots, you can judge whether the series is *stationary* or *nonstationary*. In this case, a visual inspection of the autocorrelation function plot indicates that the SALES series is nonstationary, since the ACF decays very slowly. For more formal stationarity tests, use the STATIONARITY= option. (See the section "Stationarity" on page 382.)

The inverse and partial autocorrelation plots are printed after the autocorrelation plot. These plots have the same form as the autocorrelation plots, but display inverse and partial autocorrelation values instead of autocorrelations and autocovariances. The partial and inverse autocorrelation plots are not shown in this example.

White Noise Test

The last part of the default IDENTIFY statement output is the check for white noise. This is an approximate statistical test of the hypothesis that none of the autocorrelations of the series up to a given lag are significantly different from 0. If this is true for all lags, then there is no information in the series to model, and no ARIMA model is needed for the series.

The autocorrelations are checked in groups of 6, and the number of lags checked depends on the NLAG= option. The check for white noise output is shown in Figure 11.4.

```
                         The ARIMA Procedure

                  Autocorrelation Check for White Noise

 To        Chi-             Pr >
Lag      Square     DF     ChiSq    ---------------Autocorrelations---------------

  6      426.44      6    <.0001     0.957     0.907     0.852     0.791     0.726     0.659
```

Figure 11.4. IDENTIFY Statement Check for White Noise

In this case, the white noise hypothesis is rejected very strongly, which is expected since the series is nonstationary. The *p* value for the test of the first six autocorrelations is printed as <0.0001, which means the *p* value is less than .0001.

Identification of the Differenced Series

Since the series is nonstationary, the next step is to transform it to a stationary series by differencing. That is, instead of modeling the SALES series itself, you model the change in SALES from one period to the next. To difference the SALES series, use another IDENTIFY statement and specify that the first difference of SALES be analyzed, as shown in the following statements:

```
identify var=sales(1) nlag=8;
run;
```

The second IDENTIFY statement produces the same information as the first but for the change in SALES from one period to the next rather than for the total sales in each period. The summary statistics output from this IDENTIFY statement is shown in Figure 11.5. Note that the period of differencing is given as 1, and one observation was lost through the differencing operation.

```
                              The ARIMA Procedure

                           Name of Variable = sales

             Period(s) of Differencing                           1
             Mean of Working Series                       0.660589
             Standard Deviation                           2.011543
             Number of Observations                             99
             Observation(s) eliminated by differencing           1
```

Figure 11.5. IDENTIFY Statement Output for Differenced Series

The autocorrelation plot for the differenced series is shown in Figure 11.6.

```
                              The ARIMA Procedure

                                Autocorrelations

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0     4.046306       1.00000   |                    |********************|
  1     3.351258       0.82823   |                .   |*****************   |
  2     2.390895       0.59088   |              .     |************        |
  3     1.838925       0.45447   |             .      |*********           |
  4     1.494253       0.36929   |            .       |*******.            |
  5     1.135753       0.28069   |            .       |****** .            |
  6     0.801319       0.19804   |            .       |****   .            |
  7     0.610543       0.15089   |            .       |***    .            |
  8     0.326495       0.08069   |            .       |**     .            |

                         "." marks two standard errors
```

Figure 11.6. Autocorrelations Plot for Change in SALES

The autocorrelations decrease rapidly in this plot, indicating that the change in SALES is a stationary time series.

The next step in the Box-Jenkins methodology is to examine the patterns in the autocorrelation plot to choose candidate ARMA models to the series. The partial and inverse autocorrelation function plots are also useful aids in identifying appropriate ARMA models for the series. The partial and inverse autocorrelation function plots are shown in Figure 11.7 and Figure 11.8.

```
                       The ARIMA Procedure

                     Inverse Autocorrelations

Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  1       -0.73867    |     ***************|   .                |
  2        0.36801    |                .   |*******             |
  3       -0.17538    |                ****|   .                |
  4        0.11431    |                .   |** .                |
  5       -0.15561    |                .***|   .                |
  6        0.18899    |                .   |****                |
  7       -0.15342    |                .***|   .                |
  8        0.05952    |                .   |*  .                |
```

Figure 11.7. Inverse Autocorrelation Function Plot for Change in SALES

```
                       The ARIMA Procedure

                     Partial Autocorrelations

Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  1        0.82823    |                .   |*****************   |
  2       -0.30275    |              ******|   .                |
  3        0.23722    |                .   |*****               |
  4       -0.07450    |                .  *|   .                |
  5       -0.02654    |                .  *|   .                |
  6       -0.01012    |                .   |   .                |
  7        0.04189    |                .   |*  .                |
  8       -0.17668    |                ****|   .                |
```

Figure 11.8. Partial Autocorrelation Plot for Change in SALES

In the usual Box and Jenkins approach to ARIMA modeling, the sample autocorrelation function, inverse autocorrelation function, and partial autocorrelation function are compared with the theoretical correlation functions expected from different kinds of ARMA models. This matching of theoretical autocorrelation functions of different ARMA models to the sample autocorrelation functions computed from the response series is the heart of the identification stage of Box-Jenkins modeling. Most textbooks on time series analysis discuss the theoretical autocorrelation functions for different kinds of ARMA models.

Since the input data are only a limited sample of the series, the sample autocorrelation functions computed from the input series will only approximate the true autocorrelation functions of the process generating the series. This means that the sample autocorrelation functions will not exactly match the theoretical autocorrelation functions for any ARMA model and may have a pattern similar to that of several different ARMA models.

If the series is white noise (a purely random process), then there is no need to fit a model. The check for white noise, shown in Figure 11.9, indicates that the change in sales is highly autocorrelated. Thus, an autocorrelation model, for example an AR(1) model, might be a good candidate model to fit to this process.

```
                              The ARIMA Procedure

                      Autocorrelation Check for White Noise

  To       Chi-         Pr >
 Lag     Square    DF   ChiSq   ---------------Autocorrelations--------------

   6     154.44     6  <.0001    0.828    0.591    0.454    0.369    0.281    0.198
```

Figure 11.9. IDENTIFY Statement Check for White Noise

Estimation and Diagnostic Checking Stage

The autocorrelation plots for this series, as shown in the previous section, suggest an AR(1) model for the change in SALES. You should check the diagnostic statistics to see if the AR(1) model is adequate. Other candidate models include an MA(1) model, and low-order mixed ARMA models. In this example, the AR(1) model is tried first.

Estimating an AR(1) Model

The following statements fit an AR(1) model (an autoregressive model of order 1), which predicts the change in sales as an average change, plus some fraction of the previous change, plus a random error. To estimate an AR model, you specify the order of the autoregressive model with the P= option on an ESTIMATE statement, as shown in the following statements:

```
estimate p=1;
run;
```

The ESTIMATE statement fits the model to the data and prints parameter estimates and various diagnostic statistics that indicate how well the model fits the data. The first part of the ESTIMATE statement output, the table of parameter estimates, is shown in Figure 11.10.

```
                              The ARIMA Procedure

                      Conditional Least Squares Estimation

                                    Standard                 Approx
     Parameter       Estimate         Error    t Value     Pr > |t|      Lag

     MU               0.90280       0.65984       1.37       0.1744        0
     AR1,1            0.86847       0.05485      15.83       <.0001        1
```

Figure 11.10. Parameter Estimates for AR(1) Model

The table of parameter estimates is titled “Conditional Least Squares Estimation,” which indicates the estimation method used. You can request different estimation methods with the METHOD= option.

The table of parameter estimates lists the parameters in the model; for each parameter, the table shows the estimated value and the standard error and *t* value for the estimate. The table also indicates the lag at which the parameter appears in the model.

In this case, there are two parameters in the model. The mean term is labeled MU; its estimated value is .90280. The autoregressive parameter is labeled AR1,1; this is the coefficient of the lagged value of the change in SALES, and its estimate is .86847.

The *t* values provide significance tests for the parameter estimates and indicate whether some terms in the model may be unnecessary. In this case, the *t* value for the autoregressive parameter is 15.83, so this term is highly significant. The *t* value for MU indicates that the mean term adds little to the model.

The standard error estimates are based on large sample theory. Thus, the standard errors are labeled as approximate, and the standard errors and *t* values may not be reliable in small samples.

The next part of the ESTIMATE statement output is a table of goodness-of-fit statistics, which aid in comparing this model to other models. This output is shown in Figure 11.11.

```
                              The ARIMA Procedure

                        Constant Estimate      0.118749
                        Variance Estimate       1.15794
                        Std Error Estimate     1.076076
                        AIC                    297.4469
                        SBC                    302.6372
                        Number of Residuals          99
                  * AIC and SBC do not include log determinant.



                              The ARIMA Procedure

                  * AIC and SBC do not include log determinant.
```

Figure 11.11. Goodness-of-Fit Statistics for AR(1) Model

The “Constant Estimate” is a function of the mean term MU and the autoregressive parameters. This estimate is computed only for AR or ARMA models, but not for strictly MA models. See the section “General Notation for ARIMA Models” on page 379 for an explanation of the constant estimate.

The “Variance Estimate” is the variance of the residual series, which estimates the innovation variance. The item labeled “Std Error Estimate” is the square root of the variance estimate. In general, when comparing candidate models, smaller AIC and SBC statistics indicate the better fitting model. The section The section “Estimation Details” on page 418 explains the AIC and SBC statistics.

The ESTIMATE statement next prints a table of correlations of the parameter estimates, as shown in Figure 11.12. This table can help you assess the extent to which collinearity may have influenced the results. If two parameter estimates are very highly correlated, you might consider dropping one of them from the model.

```
                              The ARIMA Procedure

                           Correlations of Parameter
                                   Estimates

                         Parameter         MU      AR1,1

                         MU             1.000      0.114
                         AR1,1          0.114      1.000
```

Figure 11.12. Correlations of the Estimates for AR(1) Model

The next part of the ESTIMATE statement output is a check of the autocorrelations of the residuals. This output has the same form as the autocorrelation check for white noise that the IDENTIFY statement prints for the response series. The autocorrelation check of residuals is shown in Figure 11.13.

```
                              The ARIMA Procedure

                        Autocorrelation Check of Residuals

  To      Chi-          Pr >
 Lag    Square   DF   ChiSq   ---------------Autocorrelations--------------

   6     19.09    5  0.0019   0.327  -0.220  -0.128   0.068  -0.002  -0.096
  12     22.90   11  0.0183   0.072   0.116  -0.042  -0.066   0.031  -0.091
  18     31.63   17  0.0167  -0.233  -0.129  -0.024   0.056  -0.014  -0.008
  24     32.83   23  0.0841   0.009  -0.057  -0.057  -0.001   0.049  -0.015
```

Figure 11.13. Check for White Noise Residuals for AR(1) Model

The χ^2 test statistics for the residuals series indicate whether the residuals are uncorrelated (white noise) or contain additional information that might be utilized by a more complex model. In this case, the test statistics reject the no-autocorrelation hypothesis at a high level of significance. (p=0.0019 for the first six lags.) This means that the residuals are not white noise, and so the AR(1) model is not a fully adequate model for this series.

The final part of the ESTIMATE statement output is a listing of the estimated model using the back shift notation. This output is shown in Figure 11.14.

```
                          The ARIMA Procedure

                       Model for variable sales

                 Estimated Mean               0.902799
                 Period(s) of Differencing           1


                        Autoregressive Factors

                    Factor 1:  1 - 0.86847 B**(1)
```

Figure 11.14. Estimated ARIMA(1, 1, 0) Model for SALES

This listing combines the differencing specification given in the IDENTIFY statement with the parameter estimates of the model for the change in sales. Since the AR(1) model is for the change in sales, the final model for sales is an ARIMA(1,1,0) model. Using B, the back shift operator, the mathematical form of the estimated model shown in this output is as follows:

$$(1 - B)sales_t = 0.902799 + \frac{1}{(1 - 0.86847B)}a_t$$

See the section "General Notation for ARIMA Models" on page 379 for further explanation of this notation.

Estimating an ARMA(1,1) Model

The IDENTIFY statement plots suggest a mixed autoregressive and moving average model, and the previous ESTIMATE statement check of residuals indicates that an AR(1) model is not sufficient. You now try estimating an ARMA(1,1) model for the change in SALES.

An ARMA(1,1) model predicts the change in SALES as an average change, plus some fraction of the previous change, plus a random error, plus some fraction of the random error in the preceding period. An ARMA(1,1) model for the change in sales is the same as an ARIMA(1,1,1) model for the level of sales.

To estimate a mixed autoregressive moving average model, you specify the order of the moving average part of the model with the Q= option on an ESTIMATE statement in addition to specifying the order of the autoregressive part with the P= option. The following statements fit an ARMA(1,1) model to the differenced SALES series:

```
   estimate p=1 q=1;
   run;
```

The parameter estimates table and goodness-of-fit statistics for this model are shown in Figure 11.15.

```
                              The ARIMA Procedure

                 * AIC and SBC do not include log determinant.



                              The ARIMA Procedure

                       Conditional Least Squares Estimation

                                Standard                 Approx
       Parameter      Estimate     Error    t Value    Pr > |t|     Lag

       MU              0.89288   0.49391       1.81      0.0738       0
       MA1,1          -0.58935   0.08988      -6.56      <.0001       1
       AR1,1           0.74755   0.07785       9.60      <.0001       1


                        Constant Estimate      0.225409
                        Variance Estimate      0.904034
                        Std Error Estimate     0.950807
                        AIC                    273.9155
                        SBC                    281.7009
                        Number of Residuals          99
                 * AIC and SBC do not include log determinant.
```

Figure 11.15. Estimated ARMA(1, 1) Model for Change in SALES

The moving average parameter estimate, labeled “MA1,1”, is -0.58935. Both the moving average and the autoregressive parameters have significant *t* values. Note that the variance estimate, AIC, and SBC are all smaller than they were for the AR(1) model, indicating that the ARMA(1,1) model fits the data better without overparameterizing.

The check for white noise residuals is shown in Figure 11.16. The χ^2 tests show that you cannot reject the hypothesis that the residuals are uncorrelated. Thus, you conclude that the ARMA(1,1) model is adequate for the change in sales series, and there is no point in trying more complex models.

```
                              The ARIMA Procedure

                          Autocorrelation Check of Residuals

  To      Chi-          Pr >
 Lag     Square    DF   ChiSq   ---------------Autocorrelations--------------

   6       3.95     4  0.4127    0.016  -0.044  -0.068   0.145   0.024  -0.094
  12       7.03    10  0.7227    0.088   0.087  -0.037  -0.075   0.051  -0.053
  18      15.41    16  0.4951   -0.221  -0.033  -0.092   0.086  -0.074  -0.005
  24      16.96    22  0.7657    0.011  -0.066  -0.022  -0.032   0.062  -0.047
```

Figure 11.16. Check for White Noise Residuals for ARMA(1, 1) Model

The output showing the form of the estimated ARIMA(1,1,1) model for SALES is shown in Figure 11.17.

```
                        The ARIMA Procedure

                     Model for variable sales

               Estimated Mean                0.892875
               Period(s) of Differencing            1


                      Autoregressive Factors

                    Factor 1:  1 - 0.74755 B**(1)


                      Moving Average Factors

                    Factor 1:  1 + 0.58935 B**(1)
```

Figure 11.17. Estimated ARIMA(1, 1, 1) Model for SALES

The estimated model shown in this output is

$$(1-B)sales_t = 0.892875 + \frac{(1+0.58935B)}{(1-0.74755B)}a_t$$

The OUTLIER statement enables you to detect whether there are changes in the time series that are not accounted for by the currently estimated model. For a long series, this task can be computationally burdensome, therefore, in general, it is better left to when a model that fits the data reasonably well has been found. Figure 11.18 shows the output of the simplest form of the statement:

```
   outlier;
```

Two possible outliers have been found for the model in question. See the "Detecting Outliers" section on page 430, Example 11.6 on page 476, and Example 11.7 on page 478 for more details on modeling in the presence of outliers.

```
                        The ARIMA Procedure

                     Outlier Detection Summary

                Maximum number searched          2
                Number found                     2
                Significance used             0.05


                          Outlier Details
                                                              Approx
                                                      Chi-     Prob>
           Obs     Type                Estimate     Square     ChiSq

            10     Additive             0.56879       4.20    0.0403
            67     Additive             0.55698       4.42    0.0355
```

Figure 11.18. Outliers for the ARIMA(1, 1, 1) Model for SALES

Since the model diagnostic tests show that all the parameter estimates are significant and the residual series is white noise, the estimation and diagnostic checking stage is complete. You can now proceed to forecasting the SALES series with this ARIMA(1,1,1) model.

Forecasting Stage

To produce the forecast, use a FORECAST statement after the ESTIMATE statement for the model you decide is best. If the last model fit were not the best, then repeat the ESTIMATE statement for the best model before using the FORECAST statement.

Suppose that the SALES series is monthly, that you wish to forecast one year ahead from the most recently available sales figure, and that the dates for the observations are given by a variable DATE in the input data set TEST. You use the following FORECAST statement:

```
forecast lead=12 interval=month id=date out=results;
run;
```

The LEAD= option specifies how many periods ahead to forecast (12 months, in this case). The ID= option specifies the ID variable used to date the observations of the SALES time series. The INTERVAL= option indicates that data are monthly and enables PROC ARIMA to extrapolate DATE values for forecast periods. The OUT= option writes the forecasts to an output data set RESULTS. See the section "OUT= Data Set" on page 432 for information on the contents of the output data set.

By default, the FORECAST statement also prints the forecast values, as shown in Figure 11.19. This output shows for each forecast period the observation number, forecast value, standard error estimate for the forecast value, and lower and upper limits for a 95% confidence interval for the forecast.

```
                          The ARIMA Procedure

                      Forecasts for variable sales

    Obs       Forecast    Std Error       95% Confidence Limits

    101       171.0320       0.9508       169.1684       172.8955
    102       174.7534       2.4168       170.0165       179.4903
    103       177.7608       3.9879       169.9445       185.5770
    104       180.2343       5.5658       169.3256       191.1430
    105       182.3088       7.1033       168.3866       196.2310
    106       184.0850       8.5789       167.2707       200.8993
    107       185.6382       9.9841       166.0698       205.2066
    108       187.0247      11.3173       164.8433       209.2061
    109       188.2866      12.5807       163.6289       212.9443
    110       189.4553      13.7784       162.4501       216.4605
    111       190.5544      14.9153       161.3209       219.7879
    112       191.6014      15.9964       160.2491       222.9538
```

Figure 11.19. Estimated ARIMA(1, 1, 1) Model for SALES

Normally, you want the forecast values stored in an output data set, and you are not interested in seeing this printed list of the forecast. You can use the NOPRINT option on the FORECAST statement to suppress this output.

Using ARIMA Procedure Statements

The IDENTIFY, ESTIMATE, and FORECAST statements are related in a hierarchy. An IDENTIFY statement brings in a time series to be modeled; several ESTIMATE statements can follow to estimate different ARIMA models for the series; for each model estimated, several FORECAST statements can be used. Thus, a FORECAST statement must be preceded at some point by an ESTIMATE statement, and an ESTIMATE statement must be preceded at some point by an IDENTIFY statement. Additional IDENTIFY statements can be used to switch to modeling a different response series or to change the degree of differencing used.

The ARIMA procedure can be used interactively in the sense that all ARIMA procedure statements can be executed any number of times without reinvoking PROC ARIMA. You can execute ARIMA procedure statements singly or in groups by following the single statement or group of statements with a RUN statement. The output for each statement or group of statements is produced when the RUN statement is entered.

A RUN statement does not terminate the PROC ARIMA step but tells the procedure to execute the statements given so far. You can end PROC ARIMA by submitting a QUIT statement, a DATA step, another PROC step, or an ENDSAS statement.

The example in the preceding section illustrates the interactive use of ARIMA procedure statements. The complete PROC ARIMA program for that example is as follows:

```
proc arima data=test;
   identify var=sales nlag=8;
   run;
   identify var=sales(1) nlag=8;
   run;
   estimate p=1;
   run;
   estimate p=1 q=1;
   run;
   forecast lead=12 interval=month id=date out=results;
   run;
quit;
```

General Notation for ARIMA Models

The order of an ARIMA (AutoRegressive Integrated Moving-Average) model is usually denoted by the notation ARIMA(p,d,q), where

p	is the order of the autoregressive part
d	is the order of the differencing

q	is the order of the moving-average process

If no differencing is done (*d* = 0), the models are usually referred to as ARMA(*p*,*q*) models. The final model in the preceding example is an ARIMA(1,1,1) model since the IDENTIFY statement specified *d* = 1, and the final ESTIMATE statement specified *p* = 1 and *q* = 1.

Notation for Pure ARIMA Models

Mathematically the pure ARIMA model is written as

$$W_t = \mu + \frac{\theta(B)}{\phi(B)} a_t$$

where

t	indexes time
W_t	is the response series Y_t or a difference of the response series
μ	is the mean term
B	is the backshift operator; that is, $BX_t = X_{t-1}$
$\phi(B)$	is the autoregressive operator, represented as a polynomial in the back shift operator: $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$
$\theta(B)$	is the moving-average operator, represented as a polynomial in the back shift operator: $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$
a_t	is the independent disturbance, also called the random error.

The series W_t is computed by the IDENTIFY statement and is the series processed by the ESTIMATE statement. Thus, W_t is either the response series Y_t or a difference of Y_t specified by the differencing operators in the IDENTIFY statement.

For simple (nonseasonal) differencing, $W_t = (1 - B)^d Y_t$. For seasonal differencing $W_t = (1 - B)^d (1 - B^s)^D Y_t$, where *d* is the degree of nonseasonal differencing, *D* is the degree of seasonal differencing, and *s* is the length of the seasonal cycle.

For example, the mathematical form of the ARIMA(1,1,1) model estimated in the preceding example is

$$(1 - B)Y_t = \mu + \frac{(1 - \theta_1 B)}{(1 - \phi_1 B)} a_t$$

Model Constant Term

The ARIMA model can also be written as

$$\phi(B)(W_t - \mu) = \theta(B) a_t$$

or

$$\phi(B)W_t = const + \theta(B)a_t$$

where

$$const = \phi(B)\mu = \mu - \phi_1\mu - \phi_2\mu - \ldots - \phi_p\mu$$

Thus, when an autoregressive operator and a mean term are both included in the model, the constant term for the model can be represented as $\phi(B)\mu$. This value is printed with the label “Constant Estimate” in the ESTIMATE statement output.

Notation for Transfer Function Models

The general ARIMA model with input series, also called the ARIMAX model, is written as

$$W_t = \mu + \sum_i \frac{\omega_i(B)}{\delta_i(B)} B^{k_i} X_{i,t} + \frac{\theta(B)}{\phi(B)} a_t$$

where

$X_{i,t}$	is the *i*th input time series or a difference of the *i*th input series at time *t*
k_i	is the pure time delay for the effect of the *i*th input series
$\omega_i(B)$	is the numerator polynomial of the transfer function for the *i*th input series
$\delta_i(B)$	is the denominator polynomial of the transfer function for the *i*th input series.

The model can also be written more compactly as

$$W_t = \mu + \sum_i \Psi_i(B) X_{i,t} + n_t$$

where

$\Psi_i(B)$	is the transfer function weights for the *i*th input series modeled as a ratio of the ω and δ polynomials: $\Psi_i(B) = (\omega_i(B)/\delta_i(B))B^{k_i}$
n_t	is the noise series: $n_t = (\theta(B)/\phi(B))a_t$

This model expresses the response series as a combination of past values of the random shocks and past values of other input series. The response series is also called the *dependent series* or *output series*. An input time series is also referred to as an *independent series* or a *predictor series*. Response variable, dependent variable, independent variable, or predictor variable are other terms often used.

Notation for Factored Models

ARIMA models are sometimes expressed in a factored form. This means that the ϕ, θ, ω, or δ polynomials are expressed as products of simpler polynomials. For example, you could express the pure ARIMA model as

$$W_t = \mu + \frac{\theta_1(B)\theta_2(B)}{\phi_1(B)\phi_2(B)}a_t$$

where $\phi_1(B)\phi_2(B) = \phi(B)$ and $\theta_1(B)\theta_2(B) = \theta(B)$.

When an ARIMA model is expressed in factored form, the order of the model is usually expressed using a factored notation also. The order of an ARIMA model expressed as the product of two factors is denoted as ARIMA(*p*,*d*,*q*)$\times$(*P*,*D*,*Q*).

Notation for Seasonal Models

ARIMA models for time series with regular seasonal fluctuations often use differencing operators and autoregressive and moving average parameters at lags that are multiples of the length of the seasonal cycle. When all the terms in an ARIMA model factor refer to lags that are a multiple of a constant *s*, the constant is factored out and suffixed to the ARIMA(*p*,*d*,*q*) notation.

Thus, the general notation for the order of a seasonal ARIMA model with both seasonal and nonseasonal factors is ARIMA(*p*,*d*,*q*)$\times$(*P*,*D*,*Q*)$_s$. The term (*p*,*d*,*q*) gives the order of the nonseasonal part of the ARIMA model; the term (*P*,*D*,*Q*)$_s$ gives the order of the seasonal part. The value of *s* is the number of observations in a seasonal cycle: 12 for monthly series, 4 for quarterly series, 7 for daily series with day-of-week effects, and so forth.

For example, the notation ARIMA(0,1,2)$\times$(0,1,1)$_{12}$ describes a seasonal ARIMA model for monthly data with the following mathematical form:

$$(1-B)(1-B^{12})Y_t = \mu + (1-\theta_{1,1}B - \theta_{1,2}B^2)(1-\theta_{2,1}B^{12})a_t$$

Stationarity

The noise (or residual) series for an ARMA model must be *stationary*, which means that both the expected values of the series and its autocovariance function are independent of time.

The standard way to check for nonstationarity is to plot the series and its autocorrelation function. You can visually examine a graph of the series over time to see if it has a visible trend or if its variability changes noticeably over time. If the series is nonstationary, its autocorrelation function will usually decay slowly.

Another way of checking for stationarity is to use the stationarity tests described in the section "Stationarity Tests" on page 416.

Most time series are nonstationary and must be transformed to a stationary series before the ARIMA modeling process can proceed. If the series has a nonstationary variance, taking the log of the series may help. You can compute the log values in a DATA step and then analyze the log values with PROC ARIMA.

If the series has a trend over time, seasonality, or some other nonstationary pattern, the usual solution is to take the difference of the series from one period to the next and then analyze this differenced series. Sometimes a series may need to be differenced more than once or differenced at lags greater than one period. (If the trend or seasonal effects are very regular, the introduction of explanatory variables may be an appropriate alternative to differencing.)

Differencing

Differencing of the response series is specified with the VAR= option of the IDENTIFY statement by placing a list of differencing periods in parentheses after the variable name. For example, to take a simple first difference of the series SALES, use the statement

```
identify var=sales(1);
```

In this example, the change in SALES from one period to the next will be analyzed.

A deterministic seasonal pattern will also cause the series to be nonstationary, since the expected value of the series will not be the same for all time periods but will be higher or lower depending on the season. When the series has a seasonal pattern, you may want to difference the series at a lag corresponding to the length of the cycle of seasons. For example, if SALES is a monthly series, the statement

```
identify var=sales(12);
```

takes a seasonal difference of SALES, so that the series analyzed is the change in SALES from its value in the same month one year ago.

To take a second difference, add another differencing period to the list. For example, the following statement takes the second difference of SALES:

```
identify var=sales(1,1);
```

That is, SALES is differenced once at lag 1 and then differenced again, also at lag 1. The statement

```
identify var=sales(2);
```

creates a 2-span difference, that is current period sales minus sales from two periods ago. The statement

```
identify var=sales(1,12);
```

takes a second-order difference of SALES, so that the series analyzed is the difference between the current period-to-period change in SALES and the change 12 periods ago. You might want to do this if the series had both a trend over time and a seasonal pattern.

There is no limit to the order of differencing and the degree of lagging for each difference.

Differencing not only affects the series used for the IDENTIFY statement output but also applies to any following ESTIMATE and FORECAST statements. ESTIMATE statements fit ARMA models to the differenced series. FORECAST statements forecast the differences and automatically sum these differences back to undo the differencing operation specified by the IDENTIFY statement, thus producing the final forecast result.

Differencing of input series is specified by the CROSSCORR= option and works just like differencing of the response series. For example, the statement

```
identify var=y(1) crosscorr=(x1(1) x2(1));
```

takes the first difference of Y, the first difference of X1, and the first difference of X2. Whenever X1 and X2 are used in INPUT= options in following ESTIMATE statements, these names refer to the differenced series.

Subset, Seasonal, and Factored ARMA Models

The simplest way to specify an ARMA model is to give the order of the AR and MA parts with the P= and Q= options. When you do this, the model has parameters for the AR and MA parts for all lags through the order specified. However, you can control the form of the ARIMA model exactly as shown in the following section.

Subset Models

You can control which lags have parameters by specifying the P= or Q= option as a list of lags in parentheses. A model like this that includes parameters for only some lags is sometimes called a *subset* or *additive model*. For example, consider the following two ESTIMATE statements:

```
identify var=sales;
estimate p=4;
estimate p=(1 4);
```

Both specify AR(4) models, but the first has parameters for lags 1, 2, 3, and 4, while the second has parameters for lags 1 and 4, with the coefficients for lags 2 and 3 constrained to 0. The mathematical form of the autoregressive models produced by these two specifications is shown in Table 11.1.

Table 11.1. Saturated versus Subset Models

Option	Autoregressive Operator
P=4	$(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3 - \phi_4 B^4)$
P=(1 4)	$(1 - \phi_1 B - \phi_4 B^4)$

Seasonal Models

One particularly useful kind of subset model is a *seasonal model*. When the response series has a seasonal pattern, the values of the series at the same time of year in previous years may be important for modeling the series. For example, if the series SALES is observed monthly, the statements

```
   identify var=sales;
   estimate p=(12);
```

model SALES as an average value plus some fraction of its deviation from this average value a year ago, plus a random error. Although this is an AR(12) model, it has only one autoregressive parameter.

Factored Models

A factored model (also referred to as a multiplicative model) represents the ARIMA model as a product of simpler ARIMA models. For example, you might model SALES as a combination of an AR(1) process reflecting short term dependencies and an AR(12) model reflecting the seasonal pattern.

It might seem that the way to do this is with the option P=(1 12), but the AR(1) process also operates in past years; you really need autoregressive parameters at lags 1, 12, and 13. You can specify a subset model with separate parameters at these lags, or you can specify a factored model that represents the model as the product of an AR(1) model and an AR(12) model. Consider the following two ESTIMATE statements:

```
   identify var=sales;
   estimate p=(1 12 13);
   estimate p=(1)(12);
```

The mathematical form of the autoregressive models produced by these two specifications are shown in Table 11.2.

Table 11.2. Subset versus Factored Models

Option	Autoregressive Operator
P=(1 12 13)	$(1 - \phi_1 B - \phi_{12} B^{12} - \phi_{13} B^{13})$
P=(1)(12)	$(1 - \phi_1 B)(1 - \phi_{12} B^{12})$

Both models fit by these two ESTIMATE statements predict SALES from its values 1, 12, and 13 periods ago, but they use different parameterizations. The first model has three parameters, whose meanings may be hard to interpret.

The factored specification P=(1)(12) represents the model as the product of two different AR models. It has only two parameters: one that corresponds to recent effects and one that represents seasonal effects. Thus the factored model is more parsimonious, and its parameter estimates are more clearly interpretable.

Input Variables and Regression with ARMA Errors

In addition to past values of the response series and past errors, you can also model the response series using the current and past values of other series, called *input series*.

Several different names are used to describe ARIMA models with input series. *Transfer function model*, *intervention model*, *interrupted time series model*, *regression model with ARMA errors*, *Box-Tiao model*, and *ARIMAX model* are all different names for ARIMA models with input series. Pankratz (1991) refers to these models as *dynamic regression*.

Using Input Series

To use input series, list the input series in a CROSSCORR= option on the IDENTIFY statement and specify how they enter the model with an INPUT= option on the ESTIMATE statement. For example, you might use a series called PRICE to help model SALES, as shown in the following statements:

```
proc arima data=a;
   identify var=sales crosscorr=price;
   estimate input=price;
   run;
```

This example performs a simple linear regression of SALES on PRICE, producing the same results as PROC REG or another SAS regression procedure. The mathematical form of the model estimated by these statements is

$$Y_t = \mu + \omega_0 X_t + a_t$$

The parameter estimates table for this example (using simulated data) is shown in Figure 11.20. The intercept parameter is labeled MU. The regression coefficient for PRICE is labeled NUM1. (See the section "Naming of Model Parameters" on page 426 for information on how parameters for input series are named.)

```
                               The ARIMA Procedure

                      Conditional Least Squares Estimation

                            Standard                 Approx
Parameter     Estimate         Error    t Value    Pr > |t|    Lag  Variable  Shift

MU           199.83602       2.99463      66.73      <.0001      0  sales         0
NUM1          -9.99299       0.02885    -346.38      <.0001      0  price         0
```

Figure 11.20. Parameter Estimates Table for Regression Model

Any number of input variables can be used in a model. For example, the following statements fit a multiple regression of SALES on PRICE and INCOME:

```
proc arima data=a;
   identify var=sales crosscorr=(price income);
   estimate input=(price income);
   run;
```

The mathematical form of the regression model estimated by these statements is

$$Y_t = \mu + \omega_1 X_{1,t} + \omega_2 X_{2,t} + a_t$$

Lagging and Differencing Input Series

You can also difference and lag the input series. For example, the following statements regress the change in SALES on the change in PRICE lagged by one period. The difference of PRICE is specified with the CROSSCORR= option and the lag of the change in PRICE is specified by the 1 $ in the INPUT= option.

```
proc arima data=a;
   identify var=sales(1) crosscorr=price(1);
   estimate input=( 1 $ price );
   run;
```

These statements estimate the model

$$(1-B)Y_t = \mu + \omega_0(1-B)X_{t-1} + a_t$$

Regression with ARMA Errors

You can combine input series with ARMA models for the errors. For example, the following statements regress SALES on INCOME and PRICE but with the error term of the regression model (called the *noise series* in ARIMA modeling terminology) assumed to be an ARMA(1,1) process.

```
proc arima data=a;
   identify var=sales crosscorr=(price income);
   estimate p=1 q=1 input=(price income);
   run;
```

These statements estimate the model

$$Y_t = \mu + \omega_1 X_{1,t} + \omega_2 X_{2,t} + \frac{(1-\theta_1 B)}{(1-\phi_1 B)} a_t$$

Stationarity and Input Series

Note that the requirement of stationarity applies to the noise series. If there are no input variables, the response series (after differencing and minus the mean term) and the noise series are the same. However, if there are inputs, the noise series is the residual after the effect of the inputs is removed.

There is no requirement that the input series be stationary. If the inputs are nonstationary, the response series will be nonstationary, even though the noise process may be stationary.

When nonstationary input series are used, you can fit the input variables first with no ARMA model for the errors and then consider the stationarity of the residuals before identifying an ARMA model for the noise part.

Identifying Regression Models with ARMA Errors

Previous sections described the ARIMA modeling identification process using the autocorrelation function plots produced by the IDENTIFY statement. This identification process does not apply when the response series depends on input variables. This is because it is the noise process for which you need to identify an ARIMA model, and when input series are involved the response series adjusted for the mean is no longer an estimate of the noise series.

However, if the input series are independent of the noise series, you can use the residuals from the regression model as an estimate of the noise series, then apply the ARIMA modeling identification process to this residual series. This assumes that the noise process is stationary.

The PLOT option on the ESTIMATE statement produces for the model residuals the same plots as the IDENTIFY statement produces for the response series. The PLOT option prints an autocorrelation function plot, an inverse autocorrelation function plot, and a partial autocorrelation function plot for the residual series.

The following statements show how the PLOT option is used to identify the ARMA(1,1) model for the noise process used in the preceding example of regression with ARMA errors:

```
proc arima data=a;
   identify var=sales crosscorr=(price income) noprint;
   estimate input=(price income) plot;
   run;
   estimate p=1 q=1 input=(price income) plot;
   run;
```

In this example, the IDENTIFY statement includes the NOPRINT option since the autocorrelation plots for the response series are not useful when you know that the response series depends on input series.

The first ESTIMATE statement fits the regression model with no model for the noise process. The PLOT option produces plots of the autocorrelation function, inverse autocorrelation function, and partial autocorrelation function for the residual series of the regression on PRICE and INCOME.

By examining the PLOT option output for the residual series, you verify that the residual series is stationary and identify an ARMA(1,1) model for the noise process. The second ESTIMATE statement fits the final model.

Although this discussion addresses regression models, the same remarks apply to identifying an ARIMA model for the noise process in models that include input series with complex transfer functions.

Intervention Models and Interrupted Time Series

One special kind of ARIMA model with input series is called an *intervention model* or *interrupted time series* model. In an intervention model, the input series is an indicator variable containing discrete values that flag the occurrence of an event affecting the response series. This event is an intervention in or an interruption of the normal evolution of the response time series, which, in the absence of the intervention, is usually assumed to be a pure ARIMA process.

Intervention models can be used both to model and forecast the response series and to analyze the impact of the intervention. When the focus is on estimating the effect of the intervention, the process is often called *intervention analysis* or *interrupted time series analysis*.

Impulse Interventions

The intervention can be a one-time event. For example, you might want to study the effect of a short-term advertising campaign on the sales of a product. In this case, the input variable has the value of 1 for the period during which the advertising campaign took place and the value 0 for all other periods. Intervention variables of this kind are sometimes called *impulse functions* or *pulse functions*.

Suppose that SALES is a monthly series, and a special advertising effort was made during the month of March 1992. The following statements estimate the effect of this intervention assuming an ARMA(1,1) model for SALES. The model is specified just like the regression model, but the intervention variable AD is constructed in the DATA step as a zero-one indicator for the month of the advertising effort.

```
data a;
   set a;
   ad = date = '1mar1992'd;
run;

proc arima data=a;
   identify var=sales crosscorr=ad;
   estimate p=1 q=1 input=ad;
run;
```

Continuing Interventions

Other interventions can be continuing, in which case the input variable flags periods before and after the intervention. For example, you might want to study the effect of a change in tax rates on some economic measure. Another example is a study of the effect of a change in speed limits on the rate of traffic fatalities. In this case, the input variable has the value 1 after the new speed limit went into effect and the value 0 before. Intervention variables of this kind are called *step functions*.

Another example is the effect of news on product demand. Suppose it was reported in July 1996 that consumption of the product prevents heart disease (or causes cancer), and SALES is consistently higher (or lower) thereafter. The following statements model the effect of this news intervention:

```
data a;
   set a;
   news = date >= '1jul1996'd;
run;

proc arima data=a;
   identify var=sales crosscorr=news;
   estimate p=1 q=1 input=news;
run;
```

Interaction Effects

You can include any number of intervention variables in the model. Intervention variables can have any pattern–impulse and continuing interventions are just two possible cases. You can mix discrete valued intervention variables and continuous regressor variables in the same model.

You can also form interaction effects by multiplying input variables and including the product variable as another input. Indeed, as long as the dependent measure forms a regular time series, you can use PROC ARIMA to fit any general linear model in conjunction with an ARMA model for the error process by using input variables that correspond to the columns of the design matrix of the linear model.

Rational Transfer Functions and Distributed Lag Models

How an input series enters the model is called its *transfer function*. Thus, ARIMA models with input series are sometimes referred to as transfer function models.

In the preceding regression and intervention model examples, the transfer function is a single scale parameter. However, you can also specify complex transfer functions composed of numerator and denominator polynomials in the backshift operator. These transfer functions operate on the input series in the same way that the ARMA specification operates on the error term.

Numerator Factors

For example, suppose you want to model the effect of PRICE on SALES as taking place gradually with the impact distributed over several past lags of PRICE. This is illustrated by the following statements:

```
proc arima data=a;
  identify var=sales crosscorr=price;
  estimate input=( (1 2 3) price );
  run;
```

These statements estimate the model

$$Y_t = \mu + (\omega_0 - \omega_1 B - \omega_2 B^2 - \omega_3 B^3)X_t + a_t$$

This example models the effect of PRICE on SALES as a linear function of the current and three most recent values of PRICE. It is equivalent to a multiple linear regression of SALES on PRICE, LAG(PRICE), LAG2(PRICE), and LAG3(PRICE).

This is an example of a transfer function with one *numerator factor*. The numerator factors for a transfer function for an input series are like the MA part of the ARMA model for the noise series.

Denominator Factors

You can also use transfer functions with *denominator factors*. The denominator factors for a transfer function for an input series are like the AR part of the ARMA model for the noise series. Denominator factors introduce exponentially weighted, infinite distributed lags into the transfer function.

To specify transfer functions with denominator factors, place the denominator factors after a slash (/) in the INPUT= option. For example, the following statements estimate the PRICE effect as an infinite distributed lag model with exponentially declining weights:

```
proc arima data=a;
  identify var=sales crosscorr=price;
  estimate input=( / (1) price );
  run;
```

The transfer function specified by these statements is as follows:

$$\frac{\omega_0}{(1 - \delta_1 B)} X_t$$

This transfer function also can be written in the following equivalent form:

$$\omega_0 \left(1 + \sum_{i=1}^{\infty} \delta_1^i B^i \right) X_t$$

This transfer function can be used with intervention inputs. When it is used with a pulse function input, the result is an intervention effect that dies out gradually over time. When it is used with a step function input, the result is an intervention effect that increases gradually to a limiting value.

Rational Transfer Functions

By combining various numerator and denominator factors in the INPUT= option, you can specify *rational transfer functions* of any complexity. To specify an input with a general rational transfer function of the form

$$\frac{\omega(B)}{\delta(B)} B^k X_t$$

use an INPUT= option in the ESTIMATE statement of the form

input=(*k* $ (ω-*lags*) / (δ-*lags*) x)

See the section "Specifying Inputs and Transfer Functions" on page 423 for more information.

Identifying Transfer Function Models

The CROSSCORR= option of the IDENTIFY statement prints sample cross-correlation functions showing the correlations between the response series and the input series at different lags. The sample cross-correlation function can be used to help identify the form of the transfer function appropriate for an input series. See textbooks on time series analysis for information on using cross-correlation functions to identify transfer function models.

For the cross-correlation function to be meaningful, the input and response series must be filtered with a prewhitening model for the input series. See the section "Prewhitening" on page 416 for more information on this issue.

Forecasting with Input Variables

To forecast a response series using an ARIMA model with inputs, you need values of the input series for the forecast periods. You can supply values for the input variables for the forecast periods in the DATA= data set, or you can have PROC ARIMA forecast the input variables.

If you do not have future values of the input variables in the input data set used by the FORECAST statement, the input series must be forecast before the ARIMA procedure can forecast the response series. If you fit an ARIMA model to each of the input series for which you need forecasts before fitting the model for the response series, the FORECAST statement automatically uses the ARIMA models for the input series to generate the needed forecasts of the inputs.

For example, suppose you want to forecast SALES for the next 12 months. In this example, the change in SALES is predicted as a function of the lagged change in PRICE, plus an ARMA(1,1) noise process. To forecast SALES using PRICE as an input, you also need to fit an ARIMA model for PRICE.

The following statements fit an AR(2) model to the change in PRICE before fitting and forecasting the model for SALES. The FORECAST statement automatically forecasts PRICE using this AR(2) model to get the future inputs needed to produce the forecast of SALES.

```
proc arima data=a;
   identify var=price(1);
   estimate p=2;
   identify var=sales(1) crosscorr=price(1);
   estimate p=1 q=1 input=price;
   forecast lead=12 interval=month id=date out=results;
run;
```

Fitting a model to the input series is also important for identifying transfer functions. (See the section "Prewhitening" on page 416 for more information.)

Input values from the DATA= data set and input values forecast by PROC ARIMA can be combined. For example, a model for SALES might have three input series: PRICE, INCOME, and TAXRATE. For the forecast, you assume that the tax rate will be unchanged. You have a forecast for INCOME from another source but only for the first few periods of the SALES forecast you want to make. You have no future values for PRICE, which needs to be forecast as in the preceding example.

In this situation, you include observations in the input data set for all forecast periods, with SALES and PRICE set to a missing value, with TAXRATE set to its last actual value, and with INCOME set to forecast values for the periods you have forecasts for and set to missing values for later periods. In the PROC ARIMA step, you estimate ARIMA models for PRICE and INCOME before estimating the model for SALES, as shown in the following statements:

```
proc arima data=a;
  identify var=price(1);
  estimate p=2;
  identify var=income(1);
  estimate p=2;
  identify var=sales(1) crosscorr=( price(1) income(1) taxrate );
  estimate p=1 q=1 input=( price income taxrate );
  forecast lead=12 interval=month id=date out=results;
  run;
```

In forecasting SALES, the ARIMA procedure uses as inputs the value of PRICE forecast by its ARIMA model, the value of TAXRATE found in the DATA= data set, and the value of INCOME found in the DATA= data set, or, when the INCOME variable is missing, the value of INCOME forecast by its ARIMA model. (Because SALES is missing for future time periods, the estimation of model parameters is not affected by the forecast values for PRICE, INCOME, or TAXRATE.)

Data Requirements

PROC ARIMA can handle time series of moderate size; there should be at least 30 observations. With 30 or fewer observations, the parameter estimates may be poor. With thousands of observations, the method requires considerable computer time and memory.

Syntax

The ARIMA procedure uses the following statements:

PROC ARIMA *options*;
  BY *variables*;
  IDENTIFY *VAR=variable options*;
  ESTIMATE *options*;
  OUTLIER *options*;
  FORECAST *options*;

Functional Summary

The statements and options controlling the ARIMA procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set	PROC ARIMA	DATA=
	IDENTIFY	DATA=
specify the output data set	PROC ARIMA	OUT=
	FORECAST	OUT=
include only forecasts in the output data set	FORECAST	NOOUTALL
write autocovariances to output data set	IDENTIFY	OUTCOV=
write parameter estimates to an output data set	ESTIMATE	OUTEST=
write correlation of parameter estimates	ESTIMATE	OUTCORR
write covariance of parameter estimates	ESTIMATE	OUTCOV
write estimated model to an output data set	ESTIMATE	OUTMODEL=
write statistics of fit to an output data set	ESTIMATE	OUTSTAT=
Options for Identifying the Series		
difference time series and plot autocorrelations	IDENTIFY	
specify response series and differencing	IDENTIFY	VAR=
specify and cross correlate input series	IDENTIFY	CROSSCORR=
center data by subtracting the mean	IDENTIFY	CENTER
exclude missing values	IDENTIFY	NOMISS
delete previous models and start fresh	IDENTIFY	CLEAR
specify the significance level for tests	IDENTIFY	ALPHA=
perform tentative ARMA order identification using the ESACF Method	IDENTIFY	ESACF
perform tentative ARMA order identification using the MINIC Method	IDENTIFY	MINIC

Description	Statement	Option
perform tentative ARMA order identification using the SCAN Method	IDENTIFY	SCAN
specify the range of autoregressive model orders for estimating the error series for the MINIC Method	IDENTIFY	PERROR=
determines the AR dimension of the SCAN, ESACF, and MINIC tables	IDENTIFY	P=
determines the MA dimension of the SCAN, ESACF, and MINIC tables	IDENTIFY	Q=
perform stationarity tests	IDENTIFY	STATIONARITY=
selection of White Noise test statistic in the presence of missing values	IDENTIFY	WHITENOISE=
Options for Defining and Estimating the Model		
specify and estimate ARIMA models	ESTIMATE	
specify autoregressive part of model	ESTIMATE	P=
specify moving average part of model	ESTIMATE	Q=
specify input variables and transfer functions	ESTIMATE	INPUT=
drop mean term from the model	ESTIMATE	NOINT
specify the estimation method	ESTIMATE	METHOD=
use alternative form for transfer functions	ESTIMATE	ALTPARM
suppress degrees-of-freedom correction in variance estimates	ESTIMATE	NODF
selection of White Noise test statistic in the presence of missing values	ESTIMATE	WHITENOISE=
Options for Outlier Detection		
specify the significance level for tests	OUTLIER	ALPHA=
identify detected outliers with variable	OUTLIER	ID=
limit the number of outliers	OUTLIER	MAXNUM=
limit the number of outliers to a percentage of the series	OUTLIER	MAXPCT=
specify the variance estimator used for testing	OUTLIER	SIGMA=
specify the type of level shifts	OUTLIER	TYPE=
Printing Control Options		
limit number of lags shown in correlation plots	IDENTIFY	NLAG=
suppress printed output for identification	IDENTIFY	NOPRINT
plot autocorrelation functions of the residuals	ESTIMATE	PLOT
print log likelihood around the estimates	ESTIMATE	GRID
control spacing for GRID option	ESTIMATE	GRIDVAL=

Description	Statement	Option
print details of the iterative estimation process	ESTIMATE	PRINTALL
suppress printed output for estimation	ESTIMATE	NOPRINT
suppress printing of the forecast values	FORECAST	NOPRINT
print the one-step forecasts and residuals	FORECAST	PRINTALL
Options to Specify Parameter Values		
specify autoregressive starting values	ESTIMATE	AR=
specify moving average starting values	ESTIMATE	MA=
specify a starting value for the mean parameter	ESTIMATE	MU=
specify starting values for transfer functions	ESTIMATE	INITVAL=
Options to Control the Iterative Estimation Process		
specify convergence criterion	ESTIMATE	CONVERGE=
specify the maximum number of iterations	ESTIMATE	MAXITER=
specify criterion for checking for singularity	ESTIMATE	SINGULAR=
suppress the iterative estimation process	ESTIMATE	NOEST
omit initial observations from objective	ESTIMATE	BACKLIM=
specify perturbation for numerical derivatives	ESTIMATE	DELTA=
omit stationarity and invertibility checks	ESTIMATE	NOSTABLE
use preliminary estimates as starting values for ML and ULS	ESTIMATE	NOLS
Options for Forecasting		
forecast the response series	FORECAST	
specify how many periods to forecast	FORECAST	LEAD=
specify the ID variable	FORECAST	ID=
specify the periodicity of the series	FORECAST	INTERVAL=
specify size of forecast confidence limits	FORECAST	ALPHA=
start forecasting before end of the input data	FORECAST	BACK=
specify the variance term used to compute forecast standard errors and confidence limits	FORECAST	SIGSQ=
control the alignment of SAS Date values	FORECAST	ALIGN=
BY Groups		
specify BY group processing	BY	

PROC ARIMA Statement

PROC ARIMA *options;*

The following options can be used in the PROC ARIMA statement:

DATA= *SAS-data-set*
specifies the name of the SAS data set containing the time series. If different DATA= specifications appear in the PROC ARIMA and IDENTIFY statements, the one in the IDENTIFY statement is used. If the DATA= option is not specified in either the PROC ARIMA or IDENTIFY statement, the most recently created SAS data set is used.

OUT= *SAS-data-set*
specifies a SAS data set to which the forecasts are output. If different OUT= specifications appear in the PROC ARIMA and FORECAST statement, the one in the FORECAST statement is used.

BY Statement

BY *variables;*

A BY statement can be used in the ARIMA procedure to process a data set in groups of observations defined by the BY variables. Note that all IDENTIFY, ESTIMATE, and FORECAST statements specified are applied to all BY groups.

Because of the need to make data-based model selections, BY-group processing is not usually done with PROC ARIMA. You usually want different models for the different series contained in different BY-groups, and the PROC ARIMA BY statement does not let you do this.

Using a BY statement imposes certain restrictions. The BY statement must appear before the first RUN statement. If a BY statement is used, the input data must come from the data set specified in the PROC statement; that is, no input data sets can be specified in IDENTIFY statements.

When a BY statement is used with PROC ARIMA, interactive processing only applies to the first BY group. Once the end of the PROC ARIMA step is reached, all ARIMA statements specified are executed again for each of the remaining BY groups in the input data set.

IDENTIFY Statement

IDENTIFY *VAR=variable options;*

The IDENTIFY statement specifies the time series to be modeled, differences the series if desired, and computes statistics to help identify models to fit. Use an IDENTIFY statement for each time series that you want to model.

If other time series are to be used as inputs in a subsequent ESTIMATE statement, they must be listed in a CROSSCORR= list in the IDENTIFY statement.

The following options are used in the IDENTIFY statement. The VAR= option is required.

ALPHA= *significance-level*

The ALPHA= option specifies the significance level for tests in the IDENTIFY statement. The default is 0.05.

CENTER

centers each time series by subtracting its sample mean. The analysis is done on the centered data. Later, when forecasts are generated, the mean is added back. Note that centering is done after differencing. The CENTER option is normally used in conjunction with the NOCONSTANT option of the ESTIMATE statement.

CLEAR

deletes all old models. This option is useful when you want to delete old models so that the input variables are not prewhitened. (See the section "Prewhitening" on page 416 for more information.)

CROSSCORR= *variable* **(***d11, d12, ..., d1k***)**
CROSSCORR= (*variable* **(***d11, d12, ..., d1k***) ...** *variable* **(***d21, d22, ..., d2k***))**

names the variables cross correlated with the response variable given by the VAR= specification.

Each variable name can be followed by a list of differencing lags in parentheses, the same as for the VAR= specification. If differencing is specified for a variable in the CROSSCORR= list, the differenced series is cross correlated with the VAR= option series, and the differenced series is used when the ESTIMATE statement INPUT= option refers to the variable.

DATA= *SAS-data-set*

specifies the input SAS data set containing the time series. If the DATA= option is omitted, the DATA= data set specified in the PROC ARIMA statement is used; if the DATA= option is omitted from the PROC ARIMA statement as well, the most recently created data set is used.

ESACF

computes the extended sample autocorrelation function and uses these estimates to tentatively identify the autoregressive and moving average orders of mixed models.

The ESACF option generates two tables. The first table displays extended sample autocorrelation estimates, and the second table displays probability values that can be used to test the significance of these estimates. The P=$(p_{min} : p_{max})$ and Q=$(q_{min} : q_{max})$ options determine the size of the table.

The autoregressive and moving average orders are tentatively identified by finding a triangular pattern in which all values are insignificant. The ARIMA procedure finds these patterns based on the IDENTIFY statement ALPHA= option and displays possible recommendations for the orders.

The following code generates an ESACF table with dimensions of p=(0:7) and q=(0:8).

```
proc arima data=test;
   identify var=x esacf p=(0:7) q=(0:8);
run;
```

See the "The ESACF Method" section on page 411 for more information.

MINIC

uses information criteria or penalty functions to provide tentative ARMA order identification. The MINIC option generates a table containing the computed information criterion associated with various ARMA model orders. The PERROR=$(p_{\epsilon,min} : p_{\epsilon,max})$ option determines the range of the autoregressive model orders used to estimate the error series. The P=$(p_{min} : p_{max})$ and Q=$(q_{min} : q_{max})$ options determine the size of the table. The ARMA orders are tentatively identified by those orders that minimize the information criterion.

The following code generates a MINIC table with default dimensions of p=(0:5) and q=(0:5) and with the error series estimated by an autoregressive model with an order, p_ϵ, that minimizes the AIC in the range from 8 to 11.

```
proc arima data=test;
   identify var=x minic perror=(8:11);
run;
```

See the "The MINIC Method" section on page 412 for more information.

NLAG= *number*

indicates the number of lags to consider in computing the autocorrelations and cross-correlations. To obtain preliminary estimates of an ARIMA(*p*,*d*,*q*) model, the NLAG= value must be at least *p*+*q*+*d*. The number of observations must be greater than or equal to the NLAG= value. The default value for NLAG= is 24 or one-fourth the number of observations, whichever is less. Even though the NLAG= value is specified, the NLAG= value can be changed according to the data set.

NOMISS

uses only the first continuous sequence of data with no missing values. By default, all observations are used.

NOPRINT

suppresses the normal printout (including the correlation plots) generated by the IDENTIFY statement.

OUTCOV= *SAS-data-set*

writes the autocovariances, autocorrelations, inverse autocorrelations, partial autocorrelations, and cross covariances to an output SAS data set. If the OUTCOV= option is not specified, no covariance output data set is created. See the section "OUTCOV= Data Set" on page 433 for more information.

P= ($p_{min} : p_{max}$**)**

see the ESACF, MINIC, and SCAN options for details.

PERROR= ($p_{\epsilon,min} : p_{\epsilon,max}$**)**

see the ESACF, MINIC, and SCAN options for details.

Q= ($q_{min} : q_{max}$**)**

see the ESACF, MINIC, and SCAN options for details.

SCAN

computes estimates of the squared canonical correlations and uses these estimates to tentatively identify the autoregressive and moving average orders of mixed models.

The SCAN option generates two tables. The first table displays squared canonical correlation estimates, and the second table displays probability values that can be used to test the significance of these estimates. The P=$(p_{min} : p_{max})$ and Q=$(q_{min} : q_{max})$ options determine the size of each table.

The autoregressive and moving average orders are tentatively identified by finding a rectangular pattern in which all values are insignificant. The ARIMA procedure finds these patterns based on the IDENTIFY statement ALPHA= option and displays possible recommendations for the orders.

The following code generates a SCAN table with default dimensions of p=(0:5) and q=(0:5). The recommended orders are based on a significance level of 0.1.

```
proc arima data=test;
   identify var=x scan alpha=0.1;
run;
```

See the "The SCAN Method" section on page 414 for more information.

STATIONARITY=

performs stationarity tests. Stationarity tests can be used to determine whether differencing terms should be included in the model specification. In each stationarity test, the autoregressive orders can be specified by a range, *test*=ar_{max}, or as a list of values, *test*=$(ar_1, .., ar_n)$, where *test* is ADF, PP, or RW. The default is (0,1,2).

See the "Stationarity Tests" section on page 416 for more information.

STATIONARITY=(ADF= *AR orders* **DLAG=** *s***)**
STATIONARITY=(DICKEY= *AR orders* **DLAG=** *s***)**

performs augmented Dickey-Fuller tests. If the DLAG=s option specified with s is greater than one, seasonal Dickey-Fuller tests are performed. The maximum allowable value of s is 12. The default value of s is one. The following code performs augmented Dickey-Fuller tests with autoregressive orders 2 and 5.

```
proc arima data=test;
   identify var=x stationarity=(adf=(2,5));
run;
```

STATIONARITY=(PP= *AR orders***)**
STATIONARITY=(PHILLIPS= *AR orders***)**

performs Phillips-Perron tests. The following code performs Augmented Phillips-Perron tests with autoregressive orders ranging from 0 to 6.

```
proc arima data=test;
   identify var=x stationarity=(pp=6);
run;
```

STATIONARITY=(RW= *AR orders***)**
STATIONARITY=(RANDOMWALK= *AR orders***)**
performs random-walk with drift tests. The following code performs random-walk with drift tests with autoregressive orders ranging from 0 to 2.

```
proc arima data=test;
   identify var=x stationarity=(rw);
run;
```

VAR= *variable*
VAR= *variable* **(** *d1, d2, ..., dk* **)**
names the variable containing the time series to analyze. The VAR= option is required.

A list of differencing lags can be placed in parentheses after the variable name to request that the series be differenced at these lags. For example, VAR=X(1) takes the first differences of X. VAR=X(1,1) requests that X be differenced twice, both times with lag 1, producing a second difference series, which is
$(X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}$.

VAR=X(2) differences X once at lag two $(X_t - X_{t-2})$.

If differencing is specified, it is the differenced series that is processed by any subsequent ESTIMATE statement.

WHITENOISE= ST | IGNOREMISS
When the series contains missing values you can use this option to choose the type of test statistic that is used in the White Noise test of the series. If WHITENOISE=IGNOREMISS the standard Ljung-Box test statistic is used. If WHITENOISE=ST a modification of this statistic suggested by Stoffer and Toloi (1992) is used. The WHITENOISE=ST is the default.

ESTIMATE Statement

ESTIMATE *options;*

The ESTIMATE statement specifies an ARMA model or transfer function model for the response variable specified in the previous IDENTIFY statement, and produces estimates of its parameters. The ESTIMATE statement also prints diagnostic information by which to check the model. Include an ESTIMATE statement for each model that you want to estimate.

Options used in the ESTIMATE statement are described in the following sections.

Options for Defining the Model and Controlling Diagnostic Statistics

The following options are used to define the model to be estimated and to control the output that is printed.

ALTPARM

specifies the alternative parameterization of the overall scale of transfer functions in the model. See the section "Alternative Model Parameterization" on page 423 for details.

INPUT= *variable*
INPUT= (*transfer-function variable* **...)**

specifies input variables and their transfer functions.

The variables used on the INPUT= option must be included in the CROSSCORR= list in the previous IDENTIFY statement. If any differencing is specified in the CROSSCORR= list, then the differenced series is used as the input to the transfer function.

The transfer function specification for an input variable is optional. If no transfer function is specified, the input variable enters the model as a simple regressor. If specified, the transfer function specification has the following syntax:

$$S\$(L_{1,1}, L_{1,2}, \ldots)(L_{2,1}, \ldots)\ldots/(L_{j,1}, \ldots)\ldots$$

Here, *S* is a shift or lag of the input variable, the terms before the slash (/) are numerator factors, and the terms after the slash (/) are denominator factors of the transfer function. All three parts are optional. See the section "Specifying Inputs and Transfer Functions" on page 423 for details.

METHOD=ML
METHOD=ULS
METHOD=CLS

specifies the estimation method to use. METHOD=ML specifies the maximum likelihood method. METHOD=ULS specifies the unconditional least-squares method. METHOD=CLS specifies the conditional least-squares method. METHOD=CLS is the default. See the "Estimation Details" section on page 418 for more information.

NOCONSTANT
NOINT

suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter μ is omitted.)

NODF

estimates the variance by dividing the error sum of squares (SSE) by the number of residuals. The default is to divide the SSE by the number of residuals minus the number of free parameters in the model.

NOPRINT

suppresses the normal printout generated by the ESTIMATE statement. If the NOPRINT option is specified for the ESTIMATE statement, then any error and warning messages are printed to the SAS log.

P= *order*
P= (*lag, ..., lag***) ... (***lag, ..., lag***)**

specifies the autoregressive part of the model. By default, no autoregressive parameters are fit.

P=(l_1, l_2, ..., l_k) defines a model with autoregressive parameters at the specified lags. P= *order* is equivalent to P=(1, 2, ..., *order*).

A concatenation of parenthesized lists specifies a factored model. For example, P=(1,2,5)(6,12) specifies the autoregressive model

$$(1 - \phi_{1,1}B - \phi_{1,2}B^2 - \phi_{1,3}B^5)(1 - \phi_{2,1}B^6 - \phi_{2,2}B^{12})$$

PLOT

plots the residual autocorrelation functions. The sample autocorrelation, the sample inverse autocorrelation, and the sample partial autocorrelation functions of the model residuals are plotted.

Q= *order*
Q= (*lag, ..., lag***) ... (***lag, ..., lag***)**

specifies the moving-average part of the model. By default, no moving-average part is included in the model.

Q=(l_1, l_2, ..., l_k) defines a model with moving-average parameters at the specified lags. Q= *order* is equivalent to Q=(1, 2, ..., *order*). A concatenation of parenthesized lists specifies a factored model. The interpretation of factors and lags is the same as for the P= option.

WHITENOISE= ST | IGNOREMISS

When the series contains missing values you can use this option to choose the type of test statistic that is used in the White Noise test of residuals. If WHITENOISE=IGNOREMISS the standard Ljung-Box test statistic is used. If WHITENOISE=ST a modification of this statistic suggested by Stoffer and Toloi (1992) is used. The WHITENOISE=ST is the default.

Options for Output Data Sets

The following options are used to store results in SAS data sets:

OUTEST= *SAS-data-set*

writes the parameter estimates to an output data set. If the OUTCORR or OUTCOV option is used, the correlations or covariances of the estimates are also written to the OUTEST= data set. See the section "OUTEST= Data Set" on page 434 for a description of the OUTEST= output data set.

OUTCORR

writes the correlations of the parameter estimates to the OUTEST= data set.

OUTCOV

writes the covariances of the parameter estimates to the OUTEST= data set.

OUTMODEL= *SAS-data-set*

writes the model and parameter estimates to an output data set. If OUTMODEL= is not specified, no model output data set is created. See "OUTMODEL= Data Set" for a description of the OUTMODEL= output data set.

OUTSTAT= *SAS-data-set*

writes the model diagnostic statistics to an output data set. If OUTSTAT= is not specified, no statistics output data set is created. See the section "OUTSTAT= Data Set" on page 438 for a description of the OUTSTAT= output data set.

Options to Specify Parameter Values

The following options enable you to specify values for the model parameters. These options can provide starting values for the estimation process, or you can specify fixed parameters for use in the FORECAST stage and suppress the estimation process with the NOEST option. By default, the ARIMA procedure finds initial parameter estimates and uses these estimates as starting values in the iterative estimation process.

If values for any parameters are specified, values for all parameters should be given. The number of values given must agree with the model specifications.

AR= *value* **...**

lists starting values for the autoregressive parameters. See the "Initial Values" section on page 424 for more information.

INITVAL= (*initializer-spec variable* **...)**

specifies starting values for the parameters in the transfer function parts of the model. See the "Initial Values" section on page 424 for more information.

MA= *value* **...**

lists starting values for the moving-average parameters. See the "Initial Values" section on page 424 for more information.

MU= *value*

specifies the MU parameter.

NOEST

uses the values specified with the AR=, MA=, INITVAL=, and MU= options as final parameter values. The estimation process is suppressed except for estimation of the residual variance. The specified parameter values are used directly by the next FORECAST statement. When NOEST is specified, standard errors, *t* values, and the correlations between estimates are displayed as 0 or missing. (The NOEST option is useful, for example, when you wish to generate forecasts corresponding to a published model.)

Options to Control the Iterative Estimation Process

The following options can be used to control the iterative process of minimizing the error sum of squares or maximizing the log likelihood function. These tuning options are not usually needed but may be useful if convergence problems arise.

BACKLIM= $-n$

omits the specified number of initial residuals from the sum of squares or likelihood function. Omitting values can be useful for suppressing transients in transfer function models that are sensitive to start-up values.

CONVERGE= *value*

specifies the convergence criterion. Convergence is assumed when the largest change in the estimate for any parameter is less that the CONVERGE= option value. If the absolute value of the parameter estimate is greater than 0.01, the relative change is used; otherwise, the absolute change in the estimate is used. The default is CONVERGE=.001.

DELTA= *value*

specifies the perturbation value for computing numerical derivatives. The default is DELTA=.001.

GRID

prints the error sum of squares (SSE) or concentrated log likelihood surface in a small grid of the parameter space around the final estimates. For each pair of parameters, the SSE is printed for the nine parameter-value combinations formed by the grid, with a center at the final estimates and with spacing given by the GRIDVAL= specification. The GRID option may help you judge whether the estimates are truly at the optimum, since the estimation process does not always converge. For models with a large number of parameters, the GRID option produces voluminous output.

GRIDVAL= *number*

controls the spacing in the grid printed by the GRID option. The default is GRIDVAL=0.005.

MAXITER= n
MAXIT= n

specifies the maximum number of iterations allowed. The default is MAXITER=50. (The default was 15 in previous releases of SAS/ETS software.)

NOLS

begins the maximum likelihood or unconditional least-squares iterations from the preliminary estimates rather than from the conditional least-squares estimates that are produced after four iterations. See the "Estimation Details" section on page 418 for more information.

NOSTABLE

specifies that the autoregressive and moving-average parameter estimates for the noise part of the model not be restricted to the stationary and invertible regions, respectively. See the section "Stationarity and Invertibility" on page 425 for more information.

PRINTALL

prints preliminary estimation results and the iterations in the final estimation process.

SINGULAR= *value*

specifies the criterion for checking singularity. If a pivot of a sweep operation is less than the SINGULAR= value, the matrix is deemed singular. Sweep operations are performed on the Jacobian matrix during final estimation and on the covariance matrix when preliminary estimates are obtained. The default is SINGULAR=1E-7.

OUTLIER Statement

OUTLIER *options;*

The OUTLIER statement can be used to detect shifts in the level of the response series that are not accounted for by the previously estimated model. An ESTIMATE statement must precede the OUTLIER statement. The following options are used in the OUTLIER statement:

TYPE = ADDITIVE
TYPE = SHIFT
TYPE = TEMP ($d_1, \ldots, d_k$ **)**
TYPE = (< ADDITIVE > < SHIFT > < TEMP ($d_1, \ldots, d_k$ **) >)**

The TYPE= option specifies the types of level shifts to search for. The default is TYPE= (ADDITIVE SHIFT), which requests searching for additive outliers and permanent level shifts. The option TEMP($d_1, \ldots, d_k$) requests searching for temporary changes in the level of durations $d_1, \ldots, d_k$. These options can also be abbreviated as AO, LS, and TC.

ALPHA= *significance-level*

The ALPHA= option specifies the significance level for tests in the OUTLIER statement. The default is 0.05.

SIGMA= ROBUST | MSE

The statistical tests performed during the outlier detection require an estimate of error variance. Using the SIGMA= option you can choose between two types of error variance estimates. SIGMA= MSE corresponds to the usual mean squared error (MSE) estimate, and SIGMA= ROBUST corresponds to a robust estimate of the error variance. The default is SIGMA= ROBUST.

MAXNUM= *number*

This option is used to limit the number of outliers to search. The default is MAXNUM= 5.

MAXPCT= *number*

This option is similar to MAXNUM= option. In the MAXPCT= option you can limit the number of outliers to search for according to a percentage of the series length. The default is MAXPCT= 2. When both of these options are specified the minimum of the two search numbers is used.

ID= *Date-Time ID variable*

This option can be used to specify a SAS Date, Time, or Datetime identification variable to label the detected outliers. This variable must be present in the input data set.

The following examples illustrate a few possibilities for the OUTLIER statement.

The most basic usage

```
outlier;
```

sets all the options to their default values, that is, it is equivalent to

```
outlier type=(ao ls) alpha=0.05 sigma=robust maxnum=5 maxpct=2;
```

The following statement requests a search for permanent level shifts and for temporary level changes of durations 6 and 12. The search is limited to at most three changes and the significance level of the underlying tests is 0.001. MSE is used as the estimate of error variance. It also requests labeling of the detected shifts using an ID variable *date*.

```
outlier type=(ls tc(6 12)) alpha=0.001 sigma=mse maxnum=3 ID=date;
```

FORECAST Statement

FORECAST *options;*

The FORECAST statement generates forecast values for a time series using the parameter estimates produced by the previous ESTIMATE statement. See the section "Forecasting Details" on page 427 for more information on calculating forecasts.

The following options can be used in the FORECAST statement:

ALIGN= *option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

ALPHA= *n*

sets the size of the forecast confidence limits. The ALPHA= value must be between 0 and 1. When you specify ALPHA=α, the upper and lower confidence limits will have a $1 - \alpha$ confidence level. The default is ALPHA=.05, which produces 95% confidence intervals. ALPHA values are rounded to the nearest hundredth.

BACK= *n*

specifies the number of observations before the end of the data that the multistep forecasts are to begin. The BACK= option value must be less than or equal to the number of observations minus the number of parameters.

The default is BACK=0, which means that the forecast starts at the end of the available data. The end of the data is the last observation for which a noise value can be calculated. If there are no input series, the end of the data is the last nonmissing value of the response time series. If there are input series, this observation can precede the last nonmissing value of the response variable, since there may be missing values for some of the input series.

ID= *variable*

names a variable in the input data set that identifies the time periods associated with the observations. The ID= variable is used in conjunction with the INTERVAL= option to extrapolate ID values from the end of the input data to identify forecast periods in the OUT= data set.

If the INTERVAL= option specifies an interval type, the ID variable must be a SAS date or datetime variable with the spacing between observations indicated by the INTERVAL= value. If the INTERVAL= option is not used, the last input value of the ID= variable is incremented by one for each forecast period to extrapolate the ID values for forecast observations.

INTERVAL= *interval*
INTERVAL= *n*

specifies the time interval between observations. See Chapter 3, "Date Intervals, Formats, and Functions," for information on valid INTERVAL= values.

The value of the INTERVAL= option is used by PROC ARIMA to extrapolate the ID values for forecast observations and to check that the input data are in order with no missing periods. See the section "Specifying Series Periodicity" on page 429 for more details.

LEAD= *n*

specifies the number of multistep forecast values to compute. For example, if LEAD=10, PROC ARIMA forecasts for ten periods beginning with the end of the input series (or earlier if BACK= is specified). It is possible to obtain fewer than the requested number of forecasts if a transfer function model is specified and insufficient data are available to compute the forecast. The default is LEAD=24.

NOOUTALL

includes only the final forecast observations in the output data set, not the one-step forecasts for the data before the forecast period.

NOPRINT

suppresses the normal printout of the forecast and associated values.

OUT= *SAS-data-set*

writes the forecast (and other values) to an output data set. If OUT= is not specified, the OUT= data set specified in the PROC ARIMA statement is used. If OUT= is also not specified in the PROC ARIMA statement, no output data set is created. See the section "OUT= Data Set" on page 432 for more information.

PRINTALL

prints the FORECAST computation throughout the whole data set. The forecast values for the data before the forecast period (specified by the BACK= option) are one-step forecasts.

SIGSQ=

specifies the variance term used in the formula for computing forecast standard errors and confidence limits. The default value is the Variance Estimate computed by the preceding ESTIMATE statement. This option is useful when you wish to generate forecast standard errors and confidence limits based on a published model. It would often be used in conjunction with the NOEST option in the preceding ESTIMATE statement.

Details

The Inverse Autocorrelation Function

The sample inverse autocorrelation function (SIACF) plays much the same role in ARIMA modeling as the sample partial autocorrelation function (SPACF) but generally indicates subset and seasonal autoregressive models better than the SPACF.

Additionally, the SIACF may be useful for detecting over-differencing. If the data come from a nonstationary or nearly nonstationary model, the SIACF has the characteristics of a noninvertible moving average. Likewise, if the data come from a model with a noninvertible moving average, then the SIACF has nonstationary characteristics and, therefore, decays slowly. In particular, if the data have been over-differenced, the SIACF looks like a SACF from a nonstationary process.

The inverse autocorrelation function is not often discussed in textbooks, so a brief description is given here. More complete discussions can be found in Cleveland (1972), Chatfield (1980), and Priestly (1981).

Let W_t be generated by the ARMA(*p*,*q*) process

$$\phi(B)W_t = \theta(B)a_t$$

where a_t is a white noise sequence. If $\theta(B)$ is invertible (that is, if θ considered as a polynomial in B has no roots less than or equal to 1 in magnitude), then the model

$$\theta(B)Z_t = \phi(B)a_t$$

is also a valid ARMA(*q*,*p*) model. This model is sometimes referred to as the dual model. The autocorrelation function (ACF) of this dual model is called the inverse autocorrelation function (IACF) of the original model.

Notice that if the original model is a pure autoregressive model, then the IACF is an ACF corresponding to a pure moving-average model. Thus, it cuts off sharply when the lag is greater than *p*; this behavior is similar to the behavior of the partial autocorrelation function (PACF).

The sample inverse autocorrelation function (SIACF) is estimated in the ARIMA procedure by the following steps. A high-order autoregressive model is fit to the data by means of the Yule-Walker equations. The order of the autoregressive model used to calculate the SIACF is the minimum of the NLAG= value and one-half the number of observations after differencing. The SIACF is then calculated as the autocorrelation function that corresponds to this autoregressive operator when treated as a moving-average operator. That is, the autoregressive coefficients are convolved with themselves and treated as autocovariances.

Under certain conditions, the sampling distribution of the SIACF can be approximated by the sampling distribution of the SACF of the dual model (Bhansali 1980). In the plots generated by ARIMA, the confidence limit marks (.) are located at $\pm 2/\sqrt{n}$. These limits bound an approximate 95% confidence interval for the hypothesis that the data are from a white noise process.

The Partial Autocorrelation Function

The approximation for a standard error for the estimated partial autocorrelation function at lag *k* is based on a null hypothesis that a pure autoregressive Gaussian process of order *k*-1 generated the time series. This standard error is $1/\sqrt{n}$ and is used to produce the approximate 95% confidence intervals depicted by the dots in the plot.

The Cross-Correlation Function

The autocorrelation, partial and inverse autocorrelation functions described in the preceding sections help when you want to model a series as a function of its past values and past random errors. When you want to include the effects of past and current values of other series in the model, the correlations of the response series and the other series must be considered.

The CROSSCORR= option on the IDENTIFY statement computes cross correlations of the VAR= series with other series and makes these series available for use as inputs in models specified by later ESTIMATE statements.

When the CROSSCORR= option is used, PROC ARIMA prints a plot of the cross-correlation function for each variable in the CROSSCORR= list. This plot is similar in format to the other correlation plots, but shows the correlation between the two series at both lags and leads. For example

```
   identify var=y crosscorr=x ...;
```

plots the cross-correlation function of Y and X, $\mathrm{Cor}(\mathrm{y_t}, \mathrm{x_{t-s}})$, for $s = -L$ to L, where L is the value of the NLAG= option. Study of the cross-correlation functions can indicate the transfer functions through which the input series should enter the model for the response series.

The cross-correlation function is computed after any specified differencing has been done. If differencing is specified for the VAR= variable or for a variable in the CROSSCORR= list, it is the differenced series that is cross correlated (and the differenced series is processed by any following ESTIMATE statement).

For example,

```
   identify var=y(1) crosscorr=x(1);
```

computes the cross correlations of the changes in Y with the changes in X. Any following ESTIMATE statement models changes in the variables rather than the variables themselves.

The ESACF Method

The **E**xtended **S**ample **A**uto**c**orrelation **F**unction (ESACF) method can tentatively identify the orders of a *stationary or nonstationary* ARMA process based on iterated least squares estimates of the autoregressive parameters. Tsay and Tiao (1984) proposed the technique, and Choi (1992) provides useful descriptions of the algorithm.

Given a stationary or nonstationary time series $\{z_t : 1 \leq t \leq n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$, with a true autoregressive order of $p + d$, and with a true moving-average order of q, you can use the ESACF method to estimate the unknown orders $p + d$ and q by analyzing the autocorrelation functions associated with filtered series of the form

$$w_t^{(m,j)} = \hat{\Phi}_{(m,j)}(B)\tilde{z}_t = \tilde{z}_t - \sum_{i=1}^{m} \hat{\phi}_i^{(m,j)} \tilde{z}_{t-i}$$

where B represents the backshift operator, where $m = p_{min}, \ldots, p_{max}$ are the autoregressive *test* orders, where $j = q_{min} + 1, \ldots, q_{max} + 1$ are the moving average *test* orders, and where $\hat{\phi}_i^{(m,j)}$ are the autoregressive parameter estimates under the assumption that the series is an ARMA(m, j) process.

For purely autoregressive models ($j = 0$), ordinary least squares (OLS) is used to consistently estimate $\hat{\phi}_i^{(m,0)}$. For ARMA models, consistent estimates are obtained by the iterated least squares recursion formula, which is initiated by the pure autoregressive estimates:

$$\hat{\phi}_i^{(m,j)} = \hat{\phi}_i^{(m+1,j-1)} - \hat{\phi}_{i-1}^{(m,j-1)} \frac{\hat{\phi}_{m+1}^{(m+1,j-1)}}{\hat{\phi}_m^{(m,j-1)}}$$

The jth lag of the sample autocorrelation function of the filtered series, $w_t^{(m,j)}$, is the *extended sample autocorrelation function*, and it is denoted as $r_{j(m)} = r_j(w^{(m,j)})$.

The standard errors of $r_{j(m)}$ are computed in the usual way using Bartlett's approximation of the variance of the sample autocorrelation function, $var(r_{j(m)}) \approx (1 + \sum_{t=1}^{j-1} r_j^2(w^{(m,j)}))$.

If the true model is an ARMA $(p + d, q)$ process, the filtered series, $w_t^{(m,j)}$, follows an MA(q) model for $j \geq q$ so that

$$r_{j(p+d)} \approx 0 \quad j > q$$

$$r_{j(p+d)} \neq 0 \quad j = q$$

Additionally, Tsay and Tiao (1984) show that the extended sample autocorrelation satisfies

$$r_{j(m)} \approx 0 \qquad\qquad j - q > m - p - d \leq 0$$

$$r_{j(m)} \neq c(m-p-d, j-q) \qquad 0 \leq j-q \leq m-p-d$$

where $c(m-p-d, j-q)$ is a nonzero constant or a continuous random variable bounded by -1 and 1.

An ESACF table is then constructed using the $r_{j(m)}$ for $m = p_{min}, \ldots, p_{max}$ and $j = q_{min}+1, \ldots, q_{max}+1$ to identify the ARMA orders (see Table 11.3). The orders are tentatively identified by finding a right (maximal) triangular pattern with vertices located at $(p+d, q)$ and $(p+d, q_{max})$ and in which all elements are insignificant (based on asymptotic normality of the autocorrelation function). The vertex $(p+d, q)$ identifies the order. Table 11.4 depicts the theoretical pattern associated with an ARMA(1,2) series.

Table 11.3. ESACF Table

	MA					
AR	0	1	2	3	·	·
0	$r_{1(0)}$	$r_{2(0)}$	$r_{3(0)}$	$r_{4(0)}$	·	·
1	$r_{1(1)}$	$r_{2(1)}$	$r_{3(1)}$	$r_{4(1)}$	·	·
2	$r_{1(2)}$	$r_{2(2)}$	$r_{3(2)}$	$r_{4(2)}$	·	·
3	$r_{1(3)}$	$r_{2(3)}$	$r_{3(3)}$	$r_{4(3)}$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Table 11.4. Theoretical ESACF Table for an ARMA(1,2) Series

	MA							
AR	0	1	2	3	4	5	6	7
0	*	X	X	X	X	X	X	X
1	*	X	0	0	0	0	0	0
2	*	X	X	0	0	0	0	0
3	*	X	X	X	0	0	0	0
4	*	X	X	X	X	0	0	0
	X = significant terms 0 = insignificant terms * = no pattern							

The MINIC Method

The **MIN**imum **I**nformation **C**riterion (MINIC) method can tentatively identify the order of a *stationary and invertible* ARMA process. Note that Hannan and Rissannen (1982) proposed this method, and Box et al. (1994) and Choi (1992) provide useful descriptions of the algorithm.

Given a stationary and invertible time series $\{z_t : 1 \leq t \leq n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$, with a true autoregressive order of p, and with a true moving-average order of q, you can use the MINIC method to compute information criteria (or penalty functions) for various autoregressive and moving average orders. The following paragraphs provide a brief description of the algorithm.

If the series is a stationary and invertible ARMA(*p*,*q*) process of the form

$$\Phi_{(p,q)}(B)\tilde{z}_t = \Theta_{(p,q)}(B)\epsilon_t$$

the error series can be approximated by a high-order AR process

$$\hat{\epsilon}_t = \hat{\Phi}_{(p_\epsilon,q)}(B)\tilde{z}_t \approx \epsilon_t$$

where the parameter estimates $\hat{\Phi}_{(p_\epsilon,q)}$ are obtained from the Yule-Walker estimates. The choice of the autoregressive order, p_ϵ, is determined by the order that minimizes the Akaike information criterion (AIC) in the range $p_{\epsilon,min} \leq p_\epsilon \leq p_{\epsilon,max}$

$$AIC(p_\epsilon,0) = \ln(\tilde{\sigma}^2_{(p_\epsilon,0)}) + 2(p_\epsilon + 0)/n$$

where

$$\tilde{\sigma}^2_{(p_\epsilon,0)} = \frac{1}{n}\sum_{t=p_\epsilon+1}^{n}\hat{\epsilon}_t^2$$

Note that Hannan and Rissannen (1982) use the Bayesian information criterion (BIC) to determine the autoregressive order used to estimate the error series. Box et al. (1994) and Choi (1992) recommend the AIC.

Once the error series has been estimated for autoregressive *test* order $m = p_{min}, \ldots, p_{max}$ and for moving-average *test* order $j = q_{min}, \ldots, q_{max}$, the OLS estimates $\hat{\Phi}_{(m,j)}$ and $\hat{\Theta}_{(m,j)}$ are computed from the regression model

$$\tilde{z}_t = \sum_{i=1}^{m}\phi_i^{(m,j)}\tilde{z}_{t-i} + \sum_{k=1}^{j}\theta_k^{(m,j)}\hat{\epsilon}_{t-k} + error$$

From the preceding parameter estimates, the BIC is then computed

$$BIC(m,j) = \ln(\tilde{\sigma}^2_{(m,j)}) + 2(m+j)\ln(n)/n$$

where

$$\tilde{\sigma}^2_{(m,j)} = \frac{1}{n}\sum_{t=t_0}^{n}\left(\tilde{z}_t - \sum_{i=1}^{m}\phi_i^{(m,j)}\tilde{z}_{t-i} + \sum_{k=1}^{j}\theta_k^{(m,j)}\hat{\epsilon}_{t-k}\right)$$

where $t_0 = p_\epsilon + \max(\mathrm{m},\mathrm{j})$.

A MINIC table is then constructed using $BIC(m, j)$ (see Table 11.5). If $p_{max} > p_{\epsilon,min}$, the preceding regression may fail due to linear dependence on the estimated error series and the mean-corrected series. Values of $BIC(m, j)$ that cannot be computed are set to missing. For large autoregressive and moving average test orders with relatively few observations, a nearly perfect fit can result. This condition can be identified by a large negative $BIC(m, j)$ value.

Table 11.5. MINIC Table

	MA					
AR	0	1	2	3	·	·
0	$BIC(0,0)$	$BIC(0,1)$	$BIC(0,2)$	$BIC(0,3)$	·	·
1	$BIC(1,0)$	$BIC(1,1)$	$BIC(1,2)$	$BIC(1,3)$	·	·
2	$BIC(2,0)$	$BIC(2,1)$	$BIC(2,2)$	$BIC(2,3)$	·	·
3	$BIC(3,0)$	$BIC(3,1)$	$BIC(3,2)$	$BIC(3,3)$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

The SCAN Method

The **S**mallest **CAN**onical (SCAN) correlation method can tentatively identify the orders of a *stationary or nonstationary* ARMA process. Tsay and Tiao (1985) proposed the technique, and Box et al. (1994) and Choi (1992) provide useful descriptions of the algorithm.

Given a stationary or nonstationary time series $\{z_t : 1 \le t \le n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$, with a true autoregressive order of $p + d$, and with a true moving-average order of q, you can use the SCAN method to analyze eigenvalues of the correlation matrix of the ARMA process. The following paragraphs provide a brief description of the algorithm.

For autoregressive *test* order $m = p_{min}, \ldots, p_{max}$ and for moving-average *test* order $j = q_{min}, \ldots, q_{max}$, perform the following steps.

1. Let $Y_{m,t} = (\tilde{z}_t, \tilde{z}_{t-1}, \ldots, \tilde{z}_{t-m})'$. Compute the following $(m+1) \times (m+1)$ matrix

$$\begin{aligned}
\hat{\beta}(m, j+1) &= \left(\sum_t Y_{m,t-j-1} Y'_{m,t-j-1}\right)^{-1} \left(\sum_t Y_{m,t-j-1} Y'_{m,t}\right) \\
\hat{\beta}^*(m, j+1) &= \left(\sum_t Y_{m,t} Y'_{m,t}\right)^{-1} \left(\sum_t Y_{m,t} Y'_{m,t-j-1}\right) \\
\hat{A}^*(m, j) &= \hat{\beta}^*(m, j+1)\hat{\beta}(m, j+1)
\end{aligned}$$

   where t ranges from $j + m + 2$ to n.

2. Find the *smallest* eigenvalue, $\hat{\lambda}^*(m, j)$, of $\hat{A}^*(m, j)$ and its corresponding *normalized* eigenvector, $\Phi_{m,j} = (1, -\phi_1^{(m,j)}, -\phi_2^{(m,j)}, \ldots, -\phi_m^{(m,j)})$. The squared canonical correlation estimate is $\hat{\lambda}^*(m, j)$.

3. Using the $\Phi_{m,j}$ as AR(m) coefficients, obtain the residuals for $t = j + m + 1$ to n, by following the formula: $w_t^{(m,j)} = \tilde{z}_t - \phi_1^{(m,j)} \tilde{z}_{t-1} - \phi_2^{(m,j)} \tilde{z}_{t-2} - \ldots - \phi_m^{(m,j)} \tilde{z}_{t-m}$.
4. From the sample autocorrelations of the residuals, $r_k(w)$, approximate the standard error of the squared canonical correlation estimate by

   $$var(\hat{\lambda}^*(m,j)^{1/2}) \approx d(m,j)/(n-m-j)$$

   where $d(m,j) = (1 + 2\sum_{i=1}^{j-1} r_k(w^{(m,j)}))$.

The test statistic to be used as an identification criterion is

$$c(m,j) = -(n-m-j)\ln(1 - \hat{\lambda}^*(m,j)/d(m,j))$$

which is asymptotically χ_1^2 if $m = p + d$ and $j \geq q$ or if $m \geq p + d$ and $j = q$. For $m > p$ and $j < q$, there is more than one theoretical zero canonical correlation between $Y_{m,t}$ and $Y_{m,t-j-1}$. Since the $\hat{\lambda}^*(m,j)$ are the smallest canonical correlations for each (m,j), the percentiles of $c(m,j)$ are less than those of a χ_1^2; therefore, Tsay and Tiao (1985) state that it is safe to assume a χ_1^2. For $m < p$ and $j < q$, no conclusions about the distribution of $c(m,j)$ are made.

A SCAN table is then constructed using $c(m,j)$ to determine which of the $\hat{\lambda}^*(m,j)$ are significantly different from zero (see Table 11.6). The ARMA orders are tentatively identified by finding a (maximal) rectangular pattern in which the $\hat{\lambda}^*(m,j)$ are insignificant for all test orders $m \geq p + d$ and $j \geq q$. There may be more than one pair of values $(p + d, q)$ that permit such a rectangular pattern. In this case, parsimony and the number of insignificant items in the rectangular pattern should help determine the model order. Table 11.7 depicts the theoretical pattern associated with an ARMA(2,2) series.

Table 11.6. SCAN Table

	MA					
AR	0	1	2	3	·	·
0	$c(0,0)$	$c(0,1)$	$c(0,2)$	$c(0,3)$	·	·
1	$c(1,0)$	$c(1,1)$	$c(1,2)$	$c(1,3)$	·	·
2	$c(2,0)$	$c(2,1)$	$c(2,2)$	$c(2,3)$	·	·
3	$c(3,0)$	$c(3,1)$	$c(3,2)$	$c(3,3)$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Table 11.7. Theoretical SCAN Table for an ARMA(2,2) Series

	MA							
AR	0	1	2	3	4	5	6	7
0	*	X	X	X	X	X	X	X
1	*	X	X	X	X	X	X	X
2	*	X	0	0	0	0	0	0
3	*	X	0	0	0	0	0	0
4	*	X	0	0	0	0	0	0
	X = significant terms 0 = insignificant terms * = no pattern							

Stationarity Tests

When a time series has a unit root, the series is nonstationary and the ordinary least squares (OLS) estimator is not normally distributed. Dickey (1976) and Dickey and Fuller (1979) studied the limiting distribution of the OLS estimator of autoregressive models for time series with a simple unit root. Dickey, Hasza, and Fuller (1984) obtained the limiting distribution for time series with seasonal unit roots. Hamilton (1994) discusses the various types of unit root testing.

For a description of Dickey-Fuller tests, refer to the section "PROBDF Function for Dickey-Fuller Tests" on page 152 in Chapter 4. Refer to Chapter 12, "The AUTOREG Procedure," for a description of Phillips-Perron tests.

The random walk with drift test recommends whether or not an integrated times series has a drift term. Hamilton (1994) discusses this test.

Prewhitening

If, as is usually the case, an input series is autocorrelated, the direct cross-correlation function between the input and response series gives a misleading indication of the relation between the input and response series.

One solution to this problem is called *prewhitening*. You first fit an ARIMA model for the input series sufficient to reduce the residuals to white noise; then, filter the input series with this model to get the white noise residual series. You then filter the response series with the same model and cross correlate the filtered response with the filtered input series.

The ARIMA procedure performs this prewhitening process automatically when you precede the IDENTIFY statement for the response series with IDENTIFY and ESTIMATE statements to fit a model for the input series. If a model with no inputs was previously fit to a variable specified by the CROSSCORR= option, then that model is used to prewhiten both the input series and the response series before the cross correlations are computed for the input series.

For example,

```
proc arima data=in;
   identify var=x;
   estimate p=1 q=1;
   identify var=y crosscorr=x;
```

Both X and Y are filtered by the ARMA(1,1) model fit to X before the cross correlations are computed.

Note that prewhitening is done to estimate the cross-correlation function; the unfiltered series are used in any subsequent ESTIMATE or FORECAST statements, and the correlation functions of Y with its own lags are computed from the unfiltered Y series. But initial values in the ESTIMATE statement are obtained with prewhitened data; therefore, the result with prewhitening can be different from the result without prewhitening.

To suppress prewhitening for all input variables, use the CLEAR option on the IDENTIFY statement to make PROC ARIMA forget all previous models.

Prewhitening and Differencing

If the VAR= and CROSSCORR= options specify differencing, the series are differenced before the prewhitening filter is applied. When the differencing lists specified on the VAR= option for an input and on the CROSSCORR= option for that input are not the same, PROC ARIMA combines the two lists so that the differencing operators used for prewhitening include all differences in either list (in the least common multiple sense).

Identifying Transfer Function Models

When identifying a transfer function model with multiple input variables, the cross-correlation functions may be misleading if the input series are correlated with each other. Any dependencies among two or more input series will confound their cross correlations with the response series.

The prewhitening technique assumes that the input variables do not depend on past values of the response variable. If there is feedback from the response variable to an input variable, as evidenced by significant cross-correlation at negative lags, both the input and the response variables need to be prewhitened before meaningful cross correlations can be computed.

PROC ARIMA cannot handle feedback models. The STATESPACE procedure is more appropriate for models with feedback.

Missing Values and Autocorrelations

To compute the sample autocorrelation function when missing values are present, PROC ARIMA uses only cross products that do not involve missing values and employs divisors that reflect the number of cross products used rather than the total length of the series. Sample partial autocorrelations and inverse autocorrelations are then computed using the sample autocorrelation function. If necessary, a taper is employed to transform the sample autocorrelations into a positive definite sequence before calculating the partial autocorrelation and inverse correlation functions. The confidence intervals produced for these functions may not be valid when there are missing values. The distributional properties for sample correlation functions are not clear for finite samples. See Dunsmuir (1984) for some asymptotic properties of the sample correlation functions.

Estimation Details

The ARIMA procedure primarily uses the computational methods outlined by Box and Jenkins. Marquardt's method is used for the nonlinear least-squares iterations. Numerical approximations of the derivatives of the sum-of-squares function are taken using a fixed delta (controlled by the DELTA= option).

The methods do not always converge successfully for a given set of data, particularly if the starting values for the parameters are not close to the least-squares estimates.

Back-forecasting

The unconditional sum of squares is computed exactly; thus, back-forecasting is not performed. Early versions of SAS/ETS software used the back-forecasting approximation and allowed a positive value of the BACKLIM= option to control the extent of the back-forecasting. In the current version, requesting a positive number of back-forecasting steps with the BACKLIM= option has no effect.

Preliminary Estimation

If an autoregressive or moving-average operator is specified with no missing lags, preliminary estimates of the parameters are computed using the autocorrelations computed in the IDENTIFY stage. Otherwise, the preliminary estimates are arbitrarily set to values that produce stable polynomials.

When preliminary estimation is not performed by PROC ARIMA, then initial values of the coefficients for any given autoregressive or moving average factor are set to 0.1 if the degree of the polynomial associated with the factor is 9 or less. Otherwise, the coefficients are determined by expanding the polynomial $(1-0.1B)$ to an appropriate power using a recursive algorithm.

These preliminary estimates are the starting values in an iterative algorithm to compute estimates of the parameters.

Estimation Methods

Maximum Likelihood

The METHOD= ML option produces maximum likelihood estimates. The likelihood function is maximized via nonlinear least squares using Marquardt's method. Maximum likelihood estimates are more expensive to compute than the conditional least-squares estimates, however, they may be preferable in some cases (Ansley and Newbold 1980; Davidson 1981).

The maximum likelihood estimates are computed as follows. Let the univariate ARMA model be

$$\phi(B)(W_t - \mu_t) = \theta(B)a_t$$

where a_t is an independent sequence of normally distributed innovations with mean 0 and variance σ^2. Here μ_t is the mean parameter μ plus the transfer function inputs. The log likelihood function can be written as follows:

$$-\frac{1}{2\sigma^2}\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x} - \frac{1}{2}\ln(|\mathbf{\Omega}|) - \frac{n}{2}\ln(\sigma^2)$$

In this equation, *n* is the number of observations, $\sigma^2\mathbf{\Omega}$ is the variance of **x** as a function of the ϕ and θ parameters, and $|\bullet|$ denotes the determinant. The vector **x** is the time series W_t minus the structural part of the model μ_t, written as a column vector, as follows:

$$\mathbf{x} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

The maximum likelihood estimate (MLE) of σ^2 is

$$s^2 = \frac{1}{n}\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x}$$

Note that the default estimator of the variance divides by $n - r$, where *r* is the number of parameters in the model, instead of by *n*. Specifying the NODF option causes a divisor of *n* to be used.

The log likelihood concentrated with respect to σ^2 can be taken up to additive constants as

$$-\frac{n}{2}\ln(\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x}) - \frac{1}{2}\ln(|\mathbf{\Omega}|)$$

Let **H** be the lower triangular matrix with positive elements on the diagonal such that $\mathbf{HH}' = \mathbf{\Omega}$. Let **e** be the vector $\mathbf{H}^{-1}\mathbf{x}$. The concentrated log likelihood with respect to σ^2 can now be written as

$$-\frac{n}{2}\ln(\mathbf{e}'\mathbf{e}) - \ln(|\mathbf{H}|)$$

or

$$-\frac{n}{2}\ln(|\mathbf{H}|^{1/n}\mathbf{e}'\mathbf{e}|\mathbf{H}|^{1/n})$$

The MLE is produced by using a Marquardt algorithm to minimize the following sum of squares:

$$|\mathbf{H}|^{1/n}\mathbf{e}'\mathbf{e}|\mathbf{H}|^{1/n}$$

The subsequent analysis of the residuals is done using **e** as the vector of residuals.

Unconditional Least Squares

The METHOD=ULS option produces unconditional least-squares estimates. The ULS method is also referred to as the *exact least-squares* (ELS) method. For METHOD=ULS, the estimates minimize

$$\sum_{t=1}^{n} \tilde{a}_t^2 = \sum_{t=1}^{n} (x_t - \mathbf{C}_t \mathbf{V}_t^{-1}(x_1, \cdots, x_{t-1})')^2$$

where $\mathbf{C}_t$ is the covariance matrix of x_t and $(x_1, \cdots, x_{t-1})$, and $\mathbf{V}_t$ is the variance matrix of $(x_1, \cdots, x_{t-1})$. In fact, $\sum_{t=1}^{n} \tilde{a}_t^2$ is the same as $\mathbf{x}'\mathbf{\Omega^{-1}}\mathbf{x}$ and, hence, $\mathbf{e}'\mathbf{e}$. Therefore, the unconditional least-squares estimates are obtained by minimizing the sum of squared residuals rather than using the log likelihood as the criterion function.

Conditional Least Squares

The METHOD=CLS option produces conditional least-squares estimates. The CLS estimates are conditional on the assumption that the past unobserved errors are equal to 0. The series x_t can be represented in terms of the previous observations, as follows:

$$x_t = a_t + \sum_{i=1}^{\infty} \pi_i x_{t-i}$$

The π weights are computed from the ratio of the ϕ and θ polynomials, as follows:

$$\frac{\phi(B)}{\theta(B)} = 1 - \sum_{i=1}^{\infty} \pi_i B^i$$

The CLS method produces estimates minimizing

$$\sum_{t=1}^{n} \hat{a}_t^2 = \sum_{t=1}^{n} (x_t - \sum_{i=1}^{\infty} \hat{\pi}_i x_{t-i})^2$$

where the unobserved past values of x_t are set to 0 and $\hat{\pi}_i$ are computed from the estimates of ϕ and θ at each iteration.

For METHOD=ULS and METHOD=ML, initial estimates are computed using the METHOD=CLS algorithm.

Start-up for Transfer Functions

When computing the noise series for transfer function and intervention models, the start-up for the transferred variable is done assuming that past values of the input series are equal to the first value of the series. The estimates are then obtained by applying least squares or maximum likelihood to the noise series. Thus, for transfer function models, the ML option does not generate the full (multivariate ARMA) maximum likelihood estimates, but it uses only the univariate likelihood function applied to the noise series.

Because PROC ARIMA uses all of the available data for the input series to generate the noise series, other start-up options for the transferred series can be implemented by prefixing an observation to the beginning of the real data. For example, if you fit a transfer function model to the variable Y with the single input X, then you can employ a start-up using 0 for the past values by prefixing to the actual data an observation with a missing value for Y and a value of 0 for X.

Information Criteria

PROC ARIMA computes and prints two information criteria, Akaike's information criterion (AIC) (Akaike 1974; Harvey 1981) and Schwarz's Bayesian criterion (SBC) (Schwarz 1978). The AIC and SBC are used to compare competing models fit to the same series. The model with the smaller information criteria is said to fit the data better. The AIC is computed as

$$-2\ln(L) + 2k$$

where *L* is the likelihood function and *k* is the number of free parameters. The SBC is computed as

$$-2\ln(L) + \ln(n)k$$

where *n* is the number of residuals that can be computed for the time series. Sometimes Schwarz's Bayesian criterion is called the Bayesian Information criterion (BIC).

If METHOD=CLS is used to do the estimation, an approximation value of *L* is used, where *L* is based on the conditional sum of squares instead of the exact sum of squares, and a Jacobian factor is left out.

Tests of Residuals

A table of test statistics for the hypothesis that the model residuals are white noise is printed as part of the ESTIMATE statement output. The chi-square statistics used in the test for lack of fit are computed using the Ljung-Box formula

$$\chi^2_m = n(n+2)\sum_{k=1}^{m}\frac{r^2_k}{(n-k)}$$

where

$$r_k = \frac{\sum_{t=1}^{n-k} a_t a_{t+k}}{\sum_{t=1}^{n} a^2_t}$$

and a_t is the residual series.

This formula has been suggested by Ljung and Box (1978) as yielding a better fit to the asymptotic chi-square distribution than the Box-Pierce Q statistic. Some simulation studies of the finite sample properties of this statistic are given by Davies, Triggs, and Newbold (1977) and by Ljung and Box (1978). When the time series has missing values, Stoffer and Toloi (1992) suggest a modification of this test statistic that has improved distributional properties over the standard Ljung-Box formula given above. When the series contains missing values this modified test statistic is used by default.

Each chi-square statistic is computed for all lags up to the indicated lag value and is not independent of the preceding chi-square values. The null hypotheses tested is that the current set of autocorrelations is white noise.

t-values

The *t* values reported in the table of parameter estimates are approximations whose accuracy depends on the validity of the model, the nature of the model, and the length of the observed series. When the length of the observed series is short and the number of estimated parameters is large with respect to the series length, the *t* approximation is usually poor. Probability values corresponding to a *t* distribution should be interpreted carefully as they may be misleading.

Cautions During Estimation

The ARIMA procedure uses a general nonlinear least-squares estimation method that can yield problematic results if your data do not fit the model. Output should be examined carefully. The GRID option can be used to ensure the validity and quality of the results. Problems you may encounter include the following:

- Preliminary moving-average estimates may not converge. Should this occur, preliminary estimates are derived as described previously in "Preliminary Estimation" on page 418. You can supply your own preliminary estimates with the ESTIMATE statement options.
- The estimates can lead to an unstable time series process, which can cause extreme forecast values or overflows in the forecast.
- The Jacobian matrix of partial derivatives may be singular; usually, this happens because not all the parameters are identifiable. Removing some of the parameters or using a longer time series may help.
- The iterative process may not converge. PROC ARIMA's estimation method stops after *n* iterations, where *n* is the value of the MAXITER= option. If an iteration does not improve the SSE, the Marquardt parameter is increased by a factor of ten until parameters that have a smaller SSE are obtained or until the limit value of the Marquardt parameter is exceeded.
- For METHOD=CLS, the estimates may converge but not to least-squares estimates. The estimates may converge to a local minimum, the numerical calculations may be distorted by data whose sum-of-squares surface is not smooth, or the minimum may lie outside the region of invertibility or stationarity.

- If the data are differenced and a moving-average model is fit, the parameter estimates may try to converge exactly on the invertibility boundary. In this case, the standard error estimates that are based on derivatives may be inaccurate.

Specifying Inputs and Transfer Functions

Input variables and transfer functions for them may be specified using the INPUT= option on the ESTIMATE statement. The variables used on the INPUT= option must be included in the CROSSCORR= list in the previous IDENTIFY statement. If any differencing is specified in the CROSSCORR= list, then the differenced variable is used as the input to the transfer function.

General Syntax of the INPUT= Option

The general syntax of the INPUT= option is

ESTIMATE . . . **INPUT=***(transfer-function variable . . .)*

The transfer function for an input variable is optional. The name of a variable by itself can be used to specify a pure regression term for the variable.

If specified, the syntax of the transfer function is

$$S\ \$\ (L_{1,1}, L_{1,2}, \ldots)(L_{2,1}, \ldots)\ldots/(L_{i,1}, L_{i,2}, \ldots)(L_{i+1,1}, \ldots)\ldots$$

S is the number of periods of time delay (lag) for this input series. Each term in parentheses specifies a polynomial factor with parameters at the lags specified by the $L_{i,j}$ values. The terms before the slash (/) are numerator factors. The terms after the slash (/) are denominator factors. All three parts are optional.

Commas can optionally be used between input specifications to make the INPUT= option more readable. The $ sign after the shift is also optional.

Except for the first numerator factor, each of the terms $L_{i,1}, L_{i,2}, \ldots, L_{i,k}$ indicates a factor of the form

$$(1 - \omega_{i,1}B^{L_{i,1}} - \omega_{i,2}B^{L_{i,2}} - \ldots - \omega_{i,k}B^{L_{i,k}})$$

The form of the first numerator factor depends on the ALTPARM option. By default, the constant 1 in the first numerator factor is replaced with a free parameter ω_0.

Alternative Model Parameterization

When the ALTPARM option is specified, the ω_0 parameter is factored out so it multiplies the entire transfer function, and the first numerator factor has the same form as the other factors.

The ALTPARM option does not materially affect the results; it just presents the results differently. Some people prefer to see the model written one way, while others prefer the alternative representation. Table 11.8 illustrates the effect of the ALTPARM option.

Table 11.8. The ALTPARM Option

INPUT= Option	ALTPARM	Model
INPUT=((1 2)(12)/(1)X);	No	$(\omega_0 - \omega_1 B - \omega_2 B^2)(1 - \omega_3 B^{12})/(1 - \delta_1 B)X_t$
	Yes	$\omega_0(1 - \omega_1 B - \omega_2 B^2)(1 - \omega_3 B^{12})/(1 - \delta_1 B)X_t$

Differencing and Input Variables

If you difference the response series and use input variables, take care that the differencing operations do not change the meaning of the model. For example, if you want to fit the model

$$Y_t = \frac{\omega_0}{(1-\delta_1 B)} X_t + \frac{(1-\theta_1 B)}{(1-B)(1-B^{12})} a_t$$

then the IDENTIFY statement must read

```
identify var=y(1,12) crosscorr=x(1,12);
estimate q=1 input=(/(1)x) noconstant;
```

If instead you specify the differencing as

```
identify var=y(1,12) crosscorr=x;
estimate q=1 input=(/(1)x) noconstant;
```

then the model being requested is

$$Y_t = \frac{\omega_0}{(1-\delta_1 B)(1-B)(1-B^{12})} X_t + \frac{(1-\theta_1 B)}{(1-B)(1-B^{12})} a_t$$

which is a very different model.

The point to remember is that a differencing operation requested for the response variable specified by the VAR= option is applied only to that variable and not to the noise term of the model.

Initial Values

The syntax for giving initial values to transfer function parameters in the INITVAL= option parallels the syntax of the INPUT= option. For each transfer function in the INPUT= option, the INITVAL= option should give an initialization specification followed by the input series name. The initialization specification for each transfer function has the form

$$C\ \$\ (V_{1,1}, V_{1,2}, \ldots)(V_{2,1}, \ldots)\ldots/(V_{i,1}, \ldots)\ldots$$

where C is the lag 0 term in the first numerator factor of the transfer function (or the overall scale factor if the ALTPARM option is specified), and $V_{i,j}$ is the coefficient of the $L_{i,j}$ element in the transfer function.

To illustrate, suppose you want to fit the model

$$Y_t = \mu + \frac{(\omega_0 - \omega_1 B - \omega_2 B^2)}{(1 - \delta_1 B - \delta_2 B^2 - \delta_3 B^3)} X_{t-3} + \frac{1}{(1 - \phi_1 B - \phi_2 B^3)} a_t$$

and start the estimation process with the initial values μ=10, ω_0=1, ω_1=.5, ω_2=.03, δ_1=.8, δ_2=-.1, δ_3=.002, ϕ_1=.1, ϕ_2=.01. (These are arbitrary values for illustration only.) You would use the following statements:

```
identify var=y crosscorr=x;
estimate p=(1,3) input=(3$(1,2)/(1,2,3)x)
         mu=10 ar=.1 .01 initval=(1$(.5,.03)/(.8,-.1,.002)x);
```

Note that the lags specified for a particular factor will be sorted, so initial values should be given in sorted order. For example, if the P= option had been entered as P=(3,1) instead of P=(1,3), the model would be the same and so would the AR= option. Sorting is done within all factors, including transfer function factors, so initial values should always be given in order of increasing lags.

Here is another illustration, showing initialization for a factored model with multiple inputs. The model is

$$\begin{aligned} Y_t = \mu \quad & + \quad \frac{\omega_{1,0}}{(1 - \delta_{1,1} B)} W_t + (\omega_{2,0} - \omega_{2,1} B) X_{t-3} \\ & + \quad \frac{1}{(1 - \phi_1 B)(1 - \phi_2 B^6 - \phi_3 B^{12})} a_t \end{aligned}$$

and the initial values are μ=10, $\omega_{1,0}$=5, $\delta_{1,1}$=.8, $\omega_{2,0}$=1, $\omega_{2,1}$=.5, ϕ_1=.1, ϕ_2=.05, and ϕ_3=.01. You would use the following statements:

```
identify var=y crosscorr=(w x);
estimate p=(1)(6,12) input=(/(1)w, 3$(1)x)
         mu=10 ar=.1 .05 .01 initval=(5$/(.8)w 1$(.5)x);
```

Stationarity and Invertibility

By default PROC ARIMA requires that the parameter estimates for the AR and MA parts of the model always remain in the stationary and invertible regions, respectively. The NOSTABLE option removes this restriction and for high-order models may save some computer time. Note that using the NOSTABLE option does not necessarily result in an unstable model being fit, since the estimates may leave the stable region for some iterations, but still ultimately converge to stable values.

Naming of Model Parameters

In the table of parameter estimates produced by the ESTIMATE statement, model parameters are referred to using the naming convention described in this section.

The parameters in the noise part of the model are named as AR*i*,*j* or MA*i*,*j*, where AR refers to autoregressive parameters and MA to moving-average parameters. The subscript *i* refers to the particular polynomial factor, and the subscript *j* refers to the *j*th term within the *i*th factor. These terms are sorted in order of increasing lag within factors, so the subscript *j* refers to the *j*th term after sorting.

When inputs are used in the model, the parameters of each transfer function are named NUM*i*,*j* and DEN*i*,*j*. The *j*th term in the *i*th factor of a numerator polynomial is named NUM*i*,*j*. The *j*th term in the *i*th factor of a denominator polynomial is named DEN*i*,*j*.

This naming process is repeated for each input variable, so if there are multiple inputs, parameters in transfer functions for different input series have the same name. The table of parameter estimates shows in the "Variable" column the input with which each parameter is associated. The parameter name shown in the "Parameter" column and the input variable name shown in the "Variable" column must be combined to fully identify transfer function parameters.

The lag 0 parameter in the first numerator factor for the first input variable is named NUM1. For subsequent input variables, the lag 0 parameter in the first numerator factor is named NUM*k*, where *k* is the position of the input variable in the INPUT= option list. If the ALTPARM option is specified, the NUM*k* parameter is replaced by an overall scale parameter named SCALE*k*.

For the mean and noise process parameters, the response series name is shown in the "Variable" column. The Lag and Shift for each parameter are also shown in the table of parameter estimates when inputs are used.

Missing Values and Estimation and Forecasting

Estimation and forecasting are carried out in the presence of missing values by forecasting the missing values with the current set of parameter estimates. The maximum likelihood algorithm employed was suggested by Jones (1980) and is used for both unconditional least-squares (ULS) and maximum likelihood (ML) estimation.

The CLS algorithm simply fills in missing values with infinite memory forecast values, computed by forecasting ahead from the nonmissing past values as far as required by the structure of the missing values. These artificial values are then employed in the nonmissing value CLS algorithm. Artificial values are updated at each iteration along with parameter estimates.

For models with input variables, embedded missing values (that is, missing values other than at the beginning or end of the series) are not generally supported. Embedded missing values in input variables are supported for the special case of a multiple regression model having ARIMA errors. A multiple regression model is specified by an INPUT= option that simply lists the input variables (possibly with lag shifts) without any numerator or denominator transfer function factors. One-step-ahead forecasts are not available for the response variable when one or more of the input variables have missing values.

When embedded missing values are present for a model with complex transfer functions, PROC ARIMA uses the first continuous nonmissing piece of each series to do the analysis. That is, PROC ARIMA skips observations at the beginning of each series until it encounters a nonmissing value and then uses the data from there until it encounters another missing value or until the end of the data is reached. This makes the current version of PROC ARIMA compatible with earlier releases that did not allow embedded missing values.

Forecasting Details

If the model has input variables, a forecast beyond the end of the data for the input variables is possible only if univariate ARIMA models have previously been fit to the input variables or future values for the input variables are included in the DATA= data set.

If input variables are used, the forecast standard errors and confidence limits of the response depend on the estimated forecast error variance of the predicted inputs. If several input series are used, the forecast errors for the inputs should be independent; otherwise, the standard errors and confidence limits for the response series will not be accurate. If future values for the input variables are included in the DATA= data set, the standard errors of the forecasts will be underestimated since these values are assumed to be known with certainty.

The forecasts are generated using forecasting equations consistent with the method used to estimate the model parameters. Thus, the estimation method specified on the ESTIMATE statement also controls the way forecasts are produced by the FORECAST statement. If METHOD=CLS is used, the forecasts are *infinite memory forecasts*, also called *conditional forecasts*. If METHOD=ULS or METHOD=ML, the forecasts are *finite memory forecasts*, also called *unconditional forecasts*. A complete description of the steps to produce the series forecasts and their standard errors using either of these methods is quite involved and only a brief explanation of the algorithm is given in the next two sections. Additional details about the finite and infinite memory forecasts can be found in Brockwell and Davis (1991). The prediction of stationary ARMA processes is explained in Chapter 5 and the prediction of nonstationary ARMA processes is given in Chapter 9.

Infinite Memory Forecasts

If METHOD=CLS is used, the forecasts are *infinite memory forecasts*, also called *conditional forecasts*. The term *conditional* is used because the forecasts are computed by assuming that the unknown values of the response series before the start of the data are equal to the mean of the series. Thus, the forecasts are conditional on this assumption.

The series x_t can be represented as

$$x_t = a_t + \sum_{i=1}^{\infty} \pi_i x_{t-i}$$

where $\phi(B)/\theta(B) = 1 - \sum_{i=1}^{\infty} \pi_i B^i$.

The *k*-step forecast of x_{t+k} is computed as

$$\hat{x}_{t+k} = \sum_{i=1}^{k-1} \hat{\pi}_i \hat{x}_{t+k-i} + \sum_{i=k}^{\infty} \hat{\pi}_i x_{t+k-i}$$

where unobserved past values of x_t are set to zero, and $\hat{\pi}_i$ is obtained from the estimated parameters $\hat{\phi}$ and $\hat{\theta}$.

Finite Memory Forecasts

For METHOD=ULS or METHOD=ML, the forecasts are *finite memory forecasts*, also called *unconditional forecasts*. For finite memory forecasts, the covariance function of the ARMA model is used to derive the best linear prediction equation.

That is, the *k*-step forecast of x_{t+k}, given $(x_1, \cdots, x_{t-1})$, is

$$\tilde{x}_{t+k} = \mathbf{C}_{k,t} \mathbf{V}_t^{-1} (x_1, \cdots, x_{t-1})'$$

where $\mathbf{C}_{k,t}$ is the covariance of x_{t+k} and $(x_1, \cdots, x_{t-1})$, and $\mathbf{V}_t$ is the covariance matrix of the vector $(x_1, \cdots, x_{t-1})$. $\mathbf{C}_{k,t}$ and $\mathbf{V}_t$ are derived from the estimated parameters.

Finite memory forecasts minimize the mean-squared error of prediction if the parameters of the ARMA model are known exactly. (In most cases, the parameters of the ARMA model are estimated, so the predictors are not true best linear forecasts.)

If the response series is differenced, the final forecast is produced by summing the forecast of the differenced series. This summation, and, thus, the forecast, is conditional on the initial values of the series. Thus, when the response series is differenced, the final forecasts are not true finite memory forecasts because they are derived assuming that the differenced series begins in a steady-state condition. Thus, they fall somewhere between finite memory and infinite memory forecasts. In practice, there is seldom any practical difference between these forecasts and true finite memory forecasts.

Forecasting Log Transformed Data

The log transformation is often used to convert time series that are nonstationary with respect to the innovation variance into stationary time series. The usual approach is to take the log of the series in a DATA step and then apply PROC ARIMA to the transformed data. A DATA step is then used to transform the forecasts of the logs back to the original units of measurement. The confidence limits are also transformed using the exponential function.

As one alternative, you can simply exponentiate the forecast series. This procedure gives a forecast for the median of the series, but the antilog of the forecast log series underpredicts the mean of the original series. If you want to predict the expected value of the series, you need to take into account the standard error of the forecast, as shown in the following example, which uses an AR(2) model to forecast the log of a series Y:

```
data in;
   set in;
   ylog = log( y );
run;

proc arima data=in;
   identify var=ylog;
   estimate p=2;
   forecast lead=10 out=out;
run;

data out;
   set out;
   y   = exp( ylog );
   l95 = exp( l95 );
   u95 = exp( u95 );
   forecast = exp( forecast + std*std/2 );
run;
```

Specifying Series Periodicity

The INTERVAL= option is used together with the ID= variable to describe the observations that make up the time series. For example, INTERVAL=MONTH specifies a monthly time series in which each observation represents one month. See Chapter 3, “Date Intervals, Formats, and Functions,” for details on the interval values supported.

The variable specified by the ID= option in the PROC ARIMA statement identifies the time periods associated with the observations. Usually, SAS date or datetime values are used for this variable. PROC ARIMA uses the ID= variable in the following ways:

- to validate the data periodicity. When the INTERVAL= option is specified, PROC ARIMA uses the ID variable to check the data and verify that successive observations have valid ID values corresponding to successive time intervals.

When the INTERVAL= option is not used, PROC ARIMA verifies that the ID values are nonmissing and in ascending order.

- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for April 1970 follows an observation for January 1970, there is a gap in the input data with two omitted observations (namely February and March 1970). A warning message is printed when a gap in the input data is found.
- to label the forecast observations in the output data set. PROC ARIMA extrapolates the values of the ID variable for the forecast observations from the ID value at the end of the input data according to the frequency specifications of the INTERVAL= option. If the INTERVAL= option is not specified, PROC ARIMA extrapolates the ID variable by incrementing the ID variable value for the last observation in the input data by 1 for each forecast period. Values of the ID variable over the range of the input data are copied to the output data set.

The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

Detecting Outliers

You can use the OUTLIER statement to detect changes in the level of the response series that are not accounted for by the estimated model. The types of changes considered are Additive Outliers (AO), Level Shifts (LS), and Temporary Changes (TC).

Let η_t be a regression variable describing some type of change in the mean response. In time series literature η_t is called a shock signature. An additive outlier at some time point s corresponds to a shock signature η_t such that $\eta_s = 1.0$ and η_t is 0.0 at all other points. Similarly a permanent level shift originating at time s has a shock signature such that η_t is 0.0 for $t < s$ and 1.0 for $t \geq s$. A temporary level shift of duration d originating at time s will have η_t equal to 1.0 between s and $s + d$ and 0.0 otherwise.

Suppose that you are estimating the ARIMA model

$$D(B)Y_t = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

where Y_t is the response series, $D(B)$ is the differencing polynomial in the backward shift operator B (possibly identity), μ_t is the transfer function input, $\phi(B)$ and $\theta(B)$ are the AR and MA polynomials, and a_t is the Gaussian white noise series.

The problem of detection of level shifts in the OUTLIER statement is formulated as a problem of sequential selection of shock signatures that improve the model in the ESTIMATE statement. This is similar to the forward selection process in the stepwise regression procedure. The selection process starts with considering shock signatures of the type specified in the TYPE= option, originating at each nonmissing measurement. This involves testing $H_0\colon \beta = 0$ versus $H_a\colon \beta \neq 0$ in the model

$$D(B)(Y_t - \beta\eta_t) = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

for each of these shock signatures. The most significant shock signature, if it also satisfies the significance criterion in ALPHA= option, is included in the model. If no significant shock signature is found then the outlier detection process stops, otherwise this augmented model, which incorporates the selected shock signature in its transfer function input, becomes the null model for the subsequent selection process. This iterative process stops if at any stage no more significant shock signatures are found or if the number of iterations exceed the maximum search number resulting due to the MAXNUM= and MAXPCT= settings. In all these iterations the parameters of the ARIMA model in the ESTIMATE statement are held fixed.

The precise details of the testing procedure for a given shock signature η_t are as follows:

The preceding testing problem is equivalent to testing $H_0\colon \beta = 0$ versus $H_a\colon \beta \neq 0$ in the following "regression with ARMA errors" model

$$N_t = \beta\zeta_t + \frac{\theta(B)}{\phi(B)}a_t$$

where $N_t = (D(B)Y_t - \mu_t)$ is the "noise" process and $\zeta_t = D(B)\eta_t$ is the "effective" shock signature.

In this setting, under H_0, $N = (N_1, N_2, \ldots, N_n)^T$ is a mean zero Gaussian vector with variance covariance matrix $\sigma^2\Sigma$. Here σ^2 is the variance of the white noise process a_t and Σ is the variance covariance matrix associated with the ARMA model. Moreover, under H_a, N has $\beta\zeta$ as the mean vector where $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_n)^T$. Additionally, the generalized least squares estimate of β and its variance is given by

$$\begin{aligned} \hat{\beta} &= \delta/\kappa \\ \mathrm{Var}(\hat{\beta}) &= \sigma^2/\kappa \end{aligned}$$

where $\delta = \zeta^T\Sigma^{-1}N$ and $\kappa = \zeta^T\Sigma^{-1}\zeta$. The test statistic $\tau^2 = \delta^2/(\sigma^2\kappa)$ is used to test the significance of β, which has an approximate chi-squared distribution with 1 degree of freedom under H_0. The type of estimate of σ^2 used in the calculation of τ^2 can be specified by the SIGMA= option. The default setting is SIGMA=ROBUST that corresponds to a robust estimate suggested in an outlier detection procedure in X-12-ARIMA, the Census Bureau's time series analysis program; refer to Findley et al. (1998) for additional information. The setting SIGMA=MSE corresponds to the usual mean squared error estimate (MSE) computed the same way as in the ESTIMATE statement with the NODF option. The robust estimate of σ^2 is computed by the formula

$$\hat{\sigma}^2 = (1.49 \times \mathrm{Median}(|\hat{a}_t|))^2$$

where $\hat{a}_t$ are the standardized residuals of the null ARIMA model.

The quantities δ and κ are efficiently computed by a method described in de Jong and Penzer (1998); refer also to Kohn and Ansley (1985).

Modeling in the Presence of Outliers

In practice, modeling and forecasting time series data in the presence of outliers is a difficult problem for several reasons. The presence of outliers can adversely affect the model identification and estimation steps. Their presence close to the end of the observation period can have a serious impact on the forecasting performance of the model. In some cases level shifts are associated with changes in the mechanism driving the observation process, and separate models may be appropriate to different sections of the data. In view of all these difficulties, diagnostic tools such as outlier detection and residual analysis are essential in any modeling process.

The following modeling strategy, which incorporates level shift detection in the familiar Box-Jenkins modeling methodology, seems to work in many cases:

1. Proceed with model identification and estimation as usual. Suppose this results in a tentative ARIMA model, say M.
2. Check for additive and permanent level shifts unaccounted for by the model M using the OUTLIER statement. In this step, unless there is evidence to justify it, the number of level shifts searched should be kept small.
3. Augment the original dataset with the regression variables corresponding to the detected outliers.
4. Include the first few of these regression variables in M, and call this model M1. Re-estimate all the parameters of M1. It is important not to include too many of these outlier variables in the model in order to avoid the danger of over-fitting.
5. Check the adequacy of M1 by examining the parameter estimates, residual analysis, and outlier detection. Refine it more if necessary.

OUT= Data Set

The output data set produced by the OUT= option of the PROC ARIMA or FORECAST statements contains the following:

- the BY variables
- the ID variable
- the variable specified by the VAR= option in the IDENTIFY statement, which contains the actual values of the response series
- FORECAST, a numeric variable containing the one-step-ahead predicted values and the multistep forecasts
- STD, a numeric variable containing the standard errors of the forecasts

- a numeric variable containing the lower confidence limits of the forecast. This variable is named L95 by default but has a different name if the ALPHA= option specifies a different size for the confidence limits.
- RESIDUAL, a numeric variable containing the differences between actual and forecast values
- a numeric variable containing the upper confidence limits of the forecast. This variable is named U95 by default but has a different name if the ALPHA= option specifies a different size for the confidence limits.

The ID variable, the BY variables, and the time series variable are the only ones copied from the input to the output data set.

Unless the NOOUTALL option is specified, the data set contains the whole time series. The FORECAST variable has the one-step forecasts (predicted values) for the input periods, followed by *n* forecast values, where *n* is the LEAD= value. The actual and RESIDUAL values are missing beyond the end of the series.

If you specify the same OUT= data set on different FORECAST statements, the latter FORECAST statements overwrite the output from the previous FORECAST statements. If you want to combine the forecasts from different FORECAST statements in the same output data set, specify the OUT= option once on the PROC ARIMA statement and omit the OUT= option on the FORECAST statements.

When a global output data set is created by the OUT= option in the PROC ARIMA statement, the variables in the OUT= data set are defined by the first FORECAST statement that is executed. The results of subsequent FORECAST statements are vertically concatenated onto the OUT= data set. Thus, if no ID variable is specified in the first FORECAST statement that is executed, no ID variable appears in the output data set, even if one is specified in a later FORECAST statement. If an ID variable is specified in the first FORECAST statement that is executed but not in a later FORECAST statement, the value of the ID variable is the same as the last value processed for the ID variable for all observations created by the later FORECAST statement. Furthermore, even if the response variable changes in subsequent FORECAST statements, the response variable name in the output data set will be that of the first response variable analyzed.

OUTCOV= Data Set

The output data set produced by the OUTCOV= option of the IDENTIFY statement contains the following variables:

- LAG, a numeric variable containing the lags corresponding to the values of the covariance variables. The values of LAG range from 0 to N for covariance functions and from -N to N for cross-covariance functions, where N is the value of the NLAG= option.
- VAR, a character variable containing the name of the variable specified by the VAR= option.

- CROSSVAR, a character variable containing the name of the variable specified in the CROSSCORR= option, which labels the different cross-covariance functions. The CROSSVAR variable is blank for the autocovariance observations. When there is no CROSSCORR= option, this variable is not created.
- N, a numeric variable containing the number of observations used to calculate the current value of the covariance or cross-covariance function.
- COV, a numeric variable containing the autocovariance or cross-covariance function values. COV contains the autocovariances of the VAR= variable when the value of the CROSSVAR variable is blank. Otherwise COV contains the cross covariances between the VAR= variable and the variable named by the CROSSVAR variable.
- CORR, a numeric variable containing the autocorrelation or cross-correlation function values. CORR contains the autocorrelations of the VAR= variable when the value of the CROSSVAR variable is blank. Otherwise CORR contains the cross correlations between the VAR= variable and the variable named by the CROSSVAR variable.
- STDERR, a numeric variable containing the standard errors of the autocorrelations. The standard error estimate is based on the hypothesis that the process generating the time series is a pure moving-average process of order LAG-1. For the cross correlations, STDERR contains the value $1/\sqrt{n}$, which approximates the standard error under the hypothesis that the two series are uncorrelated.
- INVCORR, a numeric variable containing the inverse autocorrelation function values of the VAR= variable. For cross-correlation observations, (that is, when the value of the CROSSVAR variable is not blank), INVCORR contains missing values.
- PARTCORR, a numeric variable containing the partial autocorrelation function values of the VAR= variable. For cross-correlation observations (that is, when the value of the CROSSVAR variable is not blank), PARTCORR contains missing values.

OUTEST= Data Set

PROC ARIMA writes the parameter estimates for a model to an output data set when the OUTEST= option is specified in the ESTIMATE statement. The OUTEST= data set contains the following:

- the BY variables
- _NAME_, a character variable containing the name of the parameter for the covariance or correlation observations, or blank for the observations containing the parameter estimates. (This variable is not created if neither OUTCOV nor OUTCORR is specified.)
- _TYPE_, a character variable that identifies the type of observation. A description of the _TYPE_ variable values is given below.

- variables for model parameters

The variables for the model parameters are named as follows:

ERRORVAR	This numeric variable contains the variance estimate. The _TYPE_=EST observation for this variable contains the estimated error variance, and the remaining observations are missing.
MU	This numeric variable contains values for the mean parameter for the model. (This variable is not created if NOCONSTANT is specified.)
MA*j_k*	These numeric variables contain values for the moving average parameters. The variables for moving average parameters are named MA*j_k*, where *j* is the factor number, and *k* is the index of the parameter within a factor.
AR*j_k*	These numeric variables contain values for the autoregressive parameters. The variables for autoregressive parameters are named AR*j_k*, where *j* is the factor number, and *k* is the index of the parameter within a factor.
I*j_k*	These variables contain values for the transfer function parameters. Variables for transfer function parameters are named I*j_k*, where *j* is the number of the INPUT variable associated with the transfer function component, and *k* is the number of the parameter for the particular INPUT variable. INPUT variables are numbered according to the order in which they appear in the INPUT= list.
STATUS	This variable describes the convergence status of the model. A value of 0_CONVERGED indicates that the model converged.

The value of the _TYPE_ variable for each observation indicates the kind of value contained in the variables for model parameters for the observation. The OUTEST= data set contains observations with the following _TYPE_ values:

EST	the observation contains parameter estimates
STD	the observation contains approximate standard errors of the estimates
CORR	the observation contains correlations of the estimates. OUTCORR must be specified to get these observations.
COV	the observation contains covariances of the estimates. OUTCOV must be specified to get these observations.
FACTOR	the observation contains values that identify for each parameter the factor that contains it. Negative values indicate denominator factors in transfer function models.
LAG	the observation contains values that identify the lag associated with each parameter

SHIFT	the observation contains values that identify the shift associated with the input series for the parameter

The values given for _TYPE_=FACTOR, _TYPE_=LAG, or _TYPE_=SHIFT observations enable you to reconstruct the model employed when provided with only the OUTEST= data set.

OUTEST= Examples

This section clarifies how model parameters are stored in the OUTEST= data set with two examples.

Consider the following example:

```
proc arima data=input;
   identify var=y cross=(x1 x2);
   estimate p=(1)(6) q=(1,3)(12) input=(x1 x2) outest=est;
quit;
proc print data=est;
run;
```

The model specified by these statements is

$$Y_t = \mu + \omega_{1,0}X_{1,t} + \omega_{2,0}X_{2,t} + \frac{(1-\theta_{11}B-\theta_{12}B^3)(1-\theta_{21}B^{12})}{(1-\phi_{11}B)(1-\phi_{21}B^6)}a_t$$

The OUTEST= data set contains the values shown in Table 11.9.

Table 11.9. OUTEST= Data Set for First Example

Obs	_TYPE_	Y	MU	MA1_1	MA1_2	MA2_1	AR1_1	AR2_1	I1_1	I2_1
1	EST	σ^2	μ	θ_{11}	θ_{12}	θ_{21}	ϕ_{11}	ϕ_{21}	$\omega_{1,0}$	$\omega_{2,0}$
2	STD	.	se μ	se θ_{11}	se θ_{12}	se θ_{21}	se ϕ_{11}	se ϕ_{21}	se $\omega_{1,0}$	se $\omega_{2,0}$
3	FACTOR	.	0	1	1	2	1	2	1	1
4	LAG	.	0	1	3	12	1	6	0	0
5	SHIFT	.	0	0	0	0	0	0	0	0

Note that the symbols in the rows for _TYPE_=EST and _TYPE_=STD in Table 11.9 would be numeric values in a real data set.

Next, consider the following example:

```
proc arima data=input;
   identify var=y cross=(x1(2) x2(1));
   estimate p=1 q=1 input=(2 $ (1)/(1,2)x1 1 $ /(1)x2) outest=est;
quit;
proc print data=est;
run;
```

The model specified by these statements is

$$Y_t = \mu + \frac{\omega_{10}-\omega_{11}B}{1-\delta_{11}B-\delta_{12}B^2}X_{1,t-2} + \frac{\omega_{20}}{1-\delta_{21}B}X_{2,t-1} + \frac{(1-\theta_1B)}{(1-\phi_1B)}a_t$$

The OUTEST= data set contains the values shown in Table 11.10.

Table 11.10. OUTEST= Data Set for Second Example

Obs	_TYPE_	Y	MU	MA1_1	AR1_1	I1_1	I1_2	I1_3	I1_4	I2_1	I2_2
1	EST	σ^2	μ	θ_1	ϕ_1	ω_{10}	ω_{11}	δ_{11}	δ_{12}	ω_{20}	δ_{21}
2	STD	.	se μ	se θ_1	se ϕ_1	se ω_{10}	se ω_{11}	se δ_{11}	se δ_{12}	se ω_{20}	se δ_{21}
3	FACTOR	.	0	1	1	1	1	-1	-1	1	-1
4	LAG	.	0	1	1	0	1	1	2	0	1
5	SHIFT	.	0	0	0	2	2	2	2	1	1

OUTMODEL= Data Set

The OUTMODEL= option in the ESTIMATE statement writes an output data set that enables you to reconstruct the model. The OUTMODEL= data set contains much the same information as the OUTEST= data set but in a transposed form that may be more useful for some purposes. In addition, the OUTMODEL= data set includes the differencing operators.

The OUTMODEL data set contains the following:

- the BY variables
- _NAME_, a character variable containing the name of the response or input variable for the observation.
- _TYPE_, a character variable that contains the estimation method that was employed. The value of _TYPE_ can be CLS, ULS, or ML.
- _STATUS_, a character variable that describes the convergence status of the model. A value of 0_CONVERGED indicates that the model converged.
- _PARM_, a character variable containing the name of the parameter given by the observation. _PARM_ takes on the values ERRORVAR, MU, AR, MA, NUM, DEN, and DIF.
- _VALUE_, a numeric variable containing the value of the estimate defined by the _PARM_ variable.
- _STD_, a numeric variable containing the standard error of the estimate.
- _FACTOR_, a numeric variable indicating the number of the factor to which the parameter belongs.
- _LAG_, a numeric variable containing the number of the term within the factor containing the parameter.
- _SHIFT_, a numeric variable containing the shift value for the input variable associated with the current parameter.

The values of _FACTOR_ and _LAG_ identify which particular MA, AR, NUM, or DEN parameter estimate is given by the _VALUE_ variable. The _NAME_ variable contains the response variable name for the MU, AR, or MA parameters. Otherwise, _NAME_ contains the input variable name associated with NUM or DEN parameter estimates. The _NAME_ variable contains the appropriate variable name associated with the current DIF observation as well. The _VALUE_ variable is 1 for all DIF observations, and the _LAG_ variable indicates the degree of differencing employed.

The observations contained in the OUTMODEL= data set are identified by the _PARM_ variable. A description of the values of the _PARM_ variable follows:

NUMRESID	_VALUE_ contains the number of residuals.
NPARMS	_VALUE_ contains the number of parameters in the model.
NDIFS	_VALUE_ contains the sum of the differencing lags employed for the response variable.
ERRORVAR	_VALUE_ contains the estimate of the innovation variance.
MU	_VALUE_ contains the estimate of the mean term.
AR	_VALUE_ contains the estimate of the autoregressive parameter indexed by the _FACTOR_ and _LAG_ variable values.
MA	_VALUE_ contains the estimate of a moving average parameter indexed by the _FACTOR_ and _LAG_ variable values.
NUM	_VALUE_ contains the estimate of the parameter in the numerator factor of the transfer function of the input variable indexed by the _FACTOR_, _LAG_, and _SHIFT_ variable values.
DEN	_VALUE_ contains the estimate of the parameter in the denominator factor of the transfer function of the input variable indexed by the _FACTOR_, _LAG_, and _SHIFT_ variable values.
DIF	_VALUE_ contains the difference operator defined by the difference lag given by the value in the _LAG_ variable.

OUTSTAT= Data Set

PROC ARIMA writes the diagnostic statistics for a model to an output data set when the OUTSTAT= option is specified in the ESTIMATE statement. The OUTSTAT data set contains the following:

- the BY variables.
- _TYPE_, a character variable that contains the estimation method used. _TYPE_ can have the value CLS, ULS, or ML.
- _STAT_, a character variable containing the name of the statistic given by the _VALUE_ variable in this observation. _STAT_ takes on the values AIC, SBC, LOGLIK, SSE, NUMRESID, NPARMS, NDIFS, ERRORVAR, MU, CONV, and NITER.
- _VALUE_, a numeric variable containing the value of the statistic named by the _STAT_ variable.

The observations contained in the OUTSTAT= data set are identified by the _STAT_ variable. A description of the values of the _STAT_ variable follows:

AIC	Akaike's information criterion
SBC	Schwarz's Bayesian criterion
LOGLIK	the log likelihood, if METHOD=ML or METHOD=ULS is specified
SSE	the sum of the squared residuals
NUMRESID	the number of residuals
NPARMS	the number of parameters in the model
NDIFS	the sum of the differencing lags employed for the response variable
ERRORVAR	the estimate of the innovation variance
MU	the estimate of the mean term
CONV	tells if the estimation converged
NITER	the number of iterations

Printed Output

The ARIMA procedure produces printed output for each of the IDENTIFY, ESTIMATE, and FORECAST statements. The output produced by each ARIMA statement is described in the following sections.

IDENTIFY Statement Printed Output

The printed output of the IDENTIFY statement consists of the following:

1. a table of summary statistics, including the name of the response variable, any specified periods of differencing, the mean and standard deviation of the response series after differencing, and the number of observations after differencing
2. a plot of the sample autocorrelation function for lags up to and including the NLAG= option value. Standard errors of the autocorrelations also appear to the right of the autocorrelation plot if the value of LINESIZE= option is sufficiently large. The standard errors are derived using Bartlett's approximation (Box and Jenkins 1976, p. 177). The approximation for a standard error for the estimated autocorrelation function at lag *k* is based on a null hypothesis that a pure moving-average Gaussian process of order *k*-1 generated the time series. The relative position of an approximate 95% confidence interval under this null hypothesis is indicated by the dots in the plot, while the asterisks represent the relative magnitude of the autocorrelation value.
3. a plot of the sample inverse autocorrelation function. See the section "The Inverse Autocorrelation Function" on page 409 for more information on the inverse autocorrelation function.
4. a plot of the sample partial autocorrelation function
5. a table of test statistics for the hypothesis that the series is white noise. These test statistics are the same as the tests for white noise residuals produced by the ESTIMATE statement and are described in the section "Estimation Details" on page 418.

6. if the CROSSCORR= option is used, a plot of the sample cross-correlation function for each series specified in the CROSSCORR= option. If a model was previously estimated for a variable in the CROSSCORR= list, the cross correlations for that series are computed for the prewhitened input and response series. For each input variable with a prewhitening filter, the cross-correlation report for the input series includes
   (a) a table of test statistics for the hypothesis of no cross correlation between the input and response series
   (b) the prewhitening filter used for the prewhitening transformation of the predictor and response variables
7. if the ESACF option is used, ESACF tables are printed
8. if the MINIC option is used, a MINIC table is printed
9. if the SCAN option is used, SCAN table is printed
10. if the STATIONARITY option is used, STATIONARITY tests results are printed

ESTIMATE Statement Printed Output

The printed output of the ESTIMATE statement consists of the following:

1. when the PRINTALL option is specified, the preliminary parameter estimates and an iteration history showing the sequence of parameter estimates tried during the fitting process
2. a table of parameter estimates showing the following for each parameter: the parameter name, the parameter estimate, the approximate standard error, t value, approximate probability ($Pr > |t|$), the lag for the parameter, the input variable name for the parameter, and the lag or “Shift” for the input variable
3. the estimates of the constant term, the innovation variance (Variance Estimate), the innovation standard deviation (Std Error Estimate), Akaike’s information criterion (AIC), Schwarz’s Bayesian criterion (SBC), and the number of residuals
4. the correlation matrix of the parameter estimates
5. a table of test statistics for hypothesis that the residuals of the model are white noise titled “Autocorrelation Check of Residuals”
6. if the PLOT option is specified, autocorrelation, inverse autocorrelation, and partial autocorrelation function plots of the residuals
7. if an INPUT variable has been modeled in such a way that prewhitening is performed in the IDENTIFY step, a table of test statistics titled “Crosscorrelation Check of Residuals.” The test statistic is based on the chi-square approximation suggested by Box and Jenkins (1976, pp. 395–396). The cross-correlation function is computed using the residuals from the model as one series and the prewhitened input variable as the other series.

8. if the GRID option is specified, the sum-of-squares or likelihood surface over a grid of parameter values near the final estimates
9. a summary of the estimated model showing the autoregressive factors, moving average factors, and transfer function factors in back shift notation with the estimated parameter values.

OUTLIER Statement Printed Output

The printed output of the FORECAST statement consists of the following:

1. a summary that contains the information about the maximum number of outliers searched, the number of outliers actually detected, and the significance level used in the outlier detection.
2. a table that contains the results of the outlier detection process. The outliers are listed in the order in which they are found. This table contains the following columns:

   - The "Obs" column contains the observation number of the start of the level shift.
   - If an ID= option is specified then the "Time ID" column contains the time identification labels of the start of the level shift.
   - The "Type" column lists the type of the level shift.
   - The "Estimate" column contains $\hat{\beta}$, the estimate of the regression coefficient of the shock signature.
   - The "Chi-Square" column lists the value of the test statistic τ^2.
   - The "Approx Prob > ChiSq" column lists the approximate *p*-value of the test statistic.

FORECAST Statement Printed Output

The printed output of the FORECAST statement consists of the following:

1. a summary of the estimated model
2. a table of forecasts, with columns for the observation numbers (Obs), the forecast values (Forecast), the forecast standard errors (Std Error), lower and upper limits of the approximate 95% confidence interval (95% confidence limits). The ALPHA= option can be used to change the confidence interval for forecasts. If the PRINTALL option is specified, the forecast table also includes columns for the actual values of the response series (Actual) and the residual values (Residual), and the table includes the input observations used to estimate the model.

ODS Table Names

PROC ARIMA assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 11.11. ODS Tables Produced in PROC ARIMA

ODS Table Name	Description	Option
ODS Tables Created by the IDENTIFY Statement		
ChiSqAuto	Chi-Square statistics table for autocorrelation	
ChiSqCross	Chi-Square statistics table for cross-correlations	CROSSCORR=
CorrGraph	Correlations graph	
DescStats	Descriptive Statistics	
ESACF	Extended Sample Autocorrelation Function	ESACF option
ESACFPValues	ESACF Probability Values	ESACF option
IACFGraph	Inverse autocorrelations graph	
InputDescStats	Input Descriptive Statistics	
MINIC	Minimum Information Criterion	MINIC option
PACFGraph	Partial autocorrelations graph	
SCAN	Squared Canonical Correlation Estimates	SCAN option
SCANPValues	SCAN Chi-Square[1] Probability Values	SCAN option
StationarityTests	Stationarity tests	STATIONARITY option
TentativeOrders	Tentative Order Selection	MINIC, ESACF, or SCAN option
ODS Tables Created by the ESTIMATE Statement		
ARPolynomial	Filter Equations	
ChiSqAuto	Chi-Square statistics table for autocorrelation	
ChiSqCross	Chi-Square statistics table for cross-correlations	
CorrB	Correlations of the Estimates	
DenPolynomial	Filter Equations	
FitStatistics	Fit Statistics	
IterHistory	Conditional Least Squares Estimation	METHOD=CLS
InitialAREstimates	Initial autoregressive parameter estimates	
InitialMAEstimates	Initial moving average parameter estimates	
InputDescription	Input description	
MAPolynomial	Filter Equations	
ModelDescription	Model description	
NumPolynomial	Filter Equations	
ParameterEstimates	Parameter Estimates	
PrelimEstimates	Preliminary Estimation	

Table 11.11. (continued)

ODS Table Name	Description	Option
ObjectiveGrid	Objective function grid matrix	GRID option
OptSummary	ARIMA Estimation Optimization	PRINTALL option
ODS Tables Created by the OUTLIER Statement		
OutlierDetails	Detected outliers	
ODS Tables Created by the FORECAST Statement		
Forecasts	Forecast	

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the ARIMA procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, “Statistical Graphics Using ODS.”

When the ODS GRAPHICS are in effect, the ARIMA procedure can produce a variety of plots. The main types of plots available are as follows:

- If the NOPRINT option in the IDENTIFY statement is off, the correlation plots of the dependent series are produced after the series is appropriately differenced. These include plots of the autocorrelation, partial autocorrelation, and inverse autocorrelation functions.
- If in the ESTIMATE statement the NOPRINT option is off while the PLOT option is on, the correlation plots of the model residuals are produced. These include plots of the autocorrelation, partial autocorrelation, and inverse autocorrelation functions.
- If the NOPRINT option in the FORECAST statement is off, the time series plot of the series forecasts is produced. If ID= option is used in the FORECAST statement, the ticks on the time axis use this information; otherwise the observation numbers are used as the time axis. If PRINTALL option is on, the forecast plot contains one-step-ahead forecasts as well as the multi-step-ahead forecasts.

For an example of the use of ODS GRAPHICS in PROC ARIMA, see Example 11.8.

ODS Graph Names

PROC ARIMA assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 11.12.

Table 11.12. ODS Graphics Produced by PROC ARIMA

ODS Graph Name	Plot Description	Associated Statement
ForecastPlot	Forecast Plot	FORECAST
ResidualACFPlot	Residual Autocorrelation Plot	ESTIMATE
ResidualIACFPlot	Residual Inverse Autocorrelation Plot	ESTIMATE
ResidualPACFPlot	Residual Partial Autocorrelation Plot	ESTIMATE
SeriesACFPlot	Series Autocorrelation Plot	IDENTIFY
SeriesIACFPlot	Series Inverse Autocorrelation Plot	IDENTIFY
SeriesPACFPlot	Series Partial Autocorrelation Plot	IDENTIFY

Examples

Example 11.1. Simulated IMA Model

This example illustrates the ARIMA procedure results for a case where the true model is known. An integrated moving average model is used for this illustration.

The following DATA step generates a pseudo-random sample of 100 periods from the ARIMA(0,1,1) process $u_t = u_{t-1} + a_t - .8a_{t-1}$, a_t iid $\mathrm{N}(0, 1)$.

```
title1 'Simulated IMA(1,1) Series';
data a;
  u1 = 0.9; a1 = 0;
  do i = -50 to 100;
     a = rannor( 32565 );
     u = u1 + a - .8 * a1;
     if i > 0 then output;
     a1 = a;
     u1 = u;
     end;
run;
```

The following ARIMA procedure statements identify and estimate the model.

```
proc arima data=a;
  identify var=u nlag=15;
  run;
  identify var=u(1) nlag=15;
  run;
  estimate q=1 ;
  run;
quit;
```

The results of the first IDENTIFY statement are shown in Output 11.1.1. The output shows the behavior of the sample autocorrelation function when the process is nonstationary. Note that in this case the estimated autocorrelations are not very high, even at small lags. Nonstationarity is reflected in a pattern of significant autocorrelations that do not decline quickly with increasing lag, not in the size of the autocorrelations.

Output 11.1.1. Output from the First IDENTIFY Statement

```
                           Simulated IMA(1,1) Series

                              The ARIMA Procedure

                              Name of Variable = u

                       Mean of Working Series    0.099637
                       Standard Deviation        1.115604
                       Number of Observations         100


                                Autocorrelations

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0     1.244572       1.00000   |                    |********************|
  1     0.547457       0.43988   |                .   |*********           |
  2     0.534787       0.42970   |               .    |*********           |
  3     0.569849       0.45787   |               .    |*********           |
  4     0.384428       0.30888   |              .     |******              |
  5     0.405137       0.32552   |              .     |*******             |
  6     0.253617       0.20378   |              .     |**** .              |
  7     0.321830       0.25859   |             .      |***** .             |
  8     0.363871       0.29237   |             .      |******.             |
  9     0.271180       0.21789   |             .      |****  .             |
 10     0.419208       0.33683   |             .      |*******             |
 11     0.298127       0.23954   |             .      |***** .             |
 12     0.186460       0.14982   |             .      |***   .             |
 13     0.313270       0.25171   |             .      |***** .             |
 14     0.314594       0.25277   |            .       |*****  .            |
 15     0.156329       0.12561   |            .       |***    .            |

                         "." marks two standard errors
```

Output 11.1.1. (continued)

```
                                The ARIMA Procedure

                              Inverse Autocorrelations

         Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

           1       -0.12382    |                . **|    .                |
           2       -0.17396    |                .***|    .                |
           3       -0.19966    |                ****|    .                |
           4       -0.01476    |                .   |    .                |
           5       -0.02895    |                .  *|    .                |
           6        0.20612    |                .   |****                 |
           7        0.01258    |                .   |    .                |
           8       -0.09616    |                . **|    .                |
           9        0.00025    |                .   |    .                |
          10       -0.16879    |                .***|    .                |
          11        0.05680    |                .   |*   .                |
          12        0.14306    |                .   |***.                 |
          13       -0.02466    |                .   |    .                |
          14       -0.15549    |                .***|    .                |
          15        0.08247    |                .   |**  .                |


                              Partial Autocorrelations

         Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

           1        0.43988    |                .   |*********            |
           2        0.29287    |                .   |******               |
           3        0.26499    |                .   |*****                |
           4       -0.00728    |                .   |    .                |
           5        0.06473    |                .   |*   .                |
           6       -0.09926    |                . **|    .                |
           7        0.10048    |                .   |**  .                |
           8        0.12872    |                .   |***.                 |
           9        0.03286    |                .   |*   .                |
          10        0.16034    |                .   |***.                 |
          11       -0.03794    |                .  *|    .                |
          12       -0.14469    |                .***|    .                |
          13        0.06415    |                .   |*   .                |
          14        0.15482    |                .   |***.                 |
          15       -0.10989    |                . **|    .                |
```

Output 11.1.1. (continued)

```
                                The ARIMA Procedure

                        Autocorrelation Check for White Noise

 To        Chi-          Pr >
Lag      Square    DF    ChiSq    --------------Autocorrelations--------------

  6       87.22     6    <.0001    0.440    0.430    0.458    0.309    0.326    0.204
 12      131.39    12    <.0001    0.259    0.292    0.218    0.337    0.240    0.150
```

The second IDENTIFY statement differences the series. The results of the second IDENTIFY statement are shown in Output 11.1.2. This output shows autocorrelation, inverse autocorrelation, and partial autocorrelation functions typical of MA(1) processes.

Output 11.1.2. Output from the Second IDENTIFY Statement

```
                              The ARIMA Procedure

                             Name of Variable = u

             Period(s) of Differencing                           1
             Mean of Working Series                       0.019752
             Standard Deviation                           1.160921
             Number of Observations                             99
             Observation(s) eliminated by differencing           1


                                Autocorrelations

Lag   Covariance   Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0     1.347737       1.00000   |                    |********************|
  1    -0.699404       -.51895   |          **********|   .                |
  2    -0.036142       -.02682   |               .   *|    .               |
  3     0.245093       0.18186   |               .    |****.               |
  4    -0.234167       -.17375   |               . ***|    .               |
  5     0.181778       0.13488   |               .    |*** .               |
  6    -0.184601       -.13697   |               . ***|    .               |
  7    0.0088659       0.00658   |               .    |    .               |
  8     0.146372       0.10861   |               .    |**  .               |
  9    -0.241579       -.17925   |               .****|    .               |
 10     0.240512       0.17846   |               .    |****.               |
 11     0.031005       0.02301   |              .     |     .              |
 12    -0.250954       -.18620   |              . ****|     .              |
 13     0.095295       0.07071   |              .     |*    .              |
 14     0.194110       0.14403   |              .     |***  .              |
 15    -0.219688       -.16300   |              .  ***|     .              |

                         "." marks two standard errors
```

Output 11.1.2. (continued)

```
                            The ARIMA Procedure

                          Inverse Autocorrelations

     Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

       1        0.72538    |                .   |***************     |
       2        0.48987    |                .   |**********          |
       3        0.35415    |                .   |*******             |
       4        0.34169    |                .   |*******             |
       5        0.33466    |                .   |*******             |
       6        0.34003    |                .   |*******             |
       7        0.24192    |                .   |*****               |
       8        0.12899    |                .   |***.                |
       9        0.06597    |                .   |*  .                |
      10        0.01654    |                .   |   .                |
      11        0.06434    |                .   |*  .                |
      12        0.08659    |                .   |** .                |
      13        0.02485    |                .   |   .                |
      14       -0.03545    |                .  *|   .                |
      15       -0.00113    |                .   |   .                |


                          Partial Autocorrelations

     Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

       1       -0.51895    |          **********|   .                |
       2       -0.40526    |            ********|   .                |
       3       -0.07862    |                . **|   .                |
       4       -0.14588    |                .***|   .                |
       5        0.02735    |                .   |*  .                |
       6       -0.13782    |                .***|   .                |
       7       -0.16741    |                .***|   .                |
       8       -0.06041    |                .  *|   .                |
       9       -0.18372    |                ****|   .                |
      10       -0.01478    |                .   |   .                |
      11        0.14277    |                .   |***.                |
      12       -0.04345    |                .  *|   .                |
      13       -0.19959    |                ****|   .                |
      14        0.08302    |                .   |** .                |
      15        0.00278    |                .   |   .                |
```

Output 11.1.2. (continued)

```
                            The ARIMA Procedure

                    Autocorrelation Check for White Noise

  To      Chi-          Pr >
 Lag    Square    DF   ChiSq   ---------------Autocorrelations---------------

   6     38.13     6  <.0001   -0.519  -0.027   0.182  -0.174   0.135  -0.137
  12     50.62    12  <.0001    0.007   0.109  -0.179   0.178   0.023  -0.186
```

The ESTIMATE statement fits an ARIMA(0,1,1) model to the simulated data. Note that in this case the parameter estimates are reasonably close to the values used to generate the simulated data. ($\mu = 0,\ \hat{\mu} = .02.\ \ \theta_1 = .8,\ \hat{\theta}_1 = .79.\ \ \sigma^2 = 1,\ \hat{\sigma}^2 = .82.$)

The ESTIMATE statement results are shown in Output 11.1.3.

Output 11.1.3. Output from Fitting ARIMA(0, 1, 1) Model

```
                          The ARIMA Procedure

                  Conditional Least Squares Estimation

                                 Standard                 Approx
      Parameter      Estimate       Error    t Value    Pr > |t|     Lag

      MU              0.02056     0.01972       1.04      0.2997       0
      MA1,1           0.79142     0.06474      12.22      <.0001       1


                        Constant Estimate      0.020558
                        Variance Estimate      0.819807
                        Std Error Estimate     0.905432
                        AIC                    263.2594
                        SBC                    268.4497
                        Number of Residuals          99
                   * AIC and SBC do not include log determinant.


                           Correlations of Parameter
                                   Estimates

                         Parameter         MU     MA1,1

                         MU             1.000    -0.124
                         MA1,1         -0.124     1.000


                       Autocorrelation Check of Residuals

    To      Chi-         Pr >
   Lag    Square    DF   ChiSq   ---------------Autocorrelations--------------

     6      6.48     5  0.2623  -0.033   0.030   0.153  -0.096   0.013  -0.163
    12     13.11    11  0.2862  -0.048   0.046  -0.086   0.159   0.027  -0.145
    18     20.12    17  0.2680   0.069   0.130  -0.099   0.006   0.164  -0.013
    24     24.73    23  0.3645   0.064   0.032   0.076  -0.077  -0.075   0.114


                            Model for variable u

                      Estimated Mean               0.020558
                      Period(s) of Differencing           1


                            Moving Average Factors

                        Factor 1:  1 - 0.79142 B**(1)
```

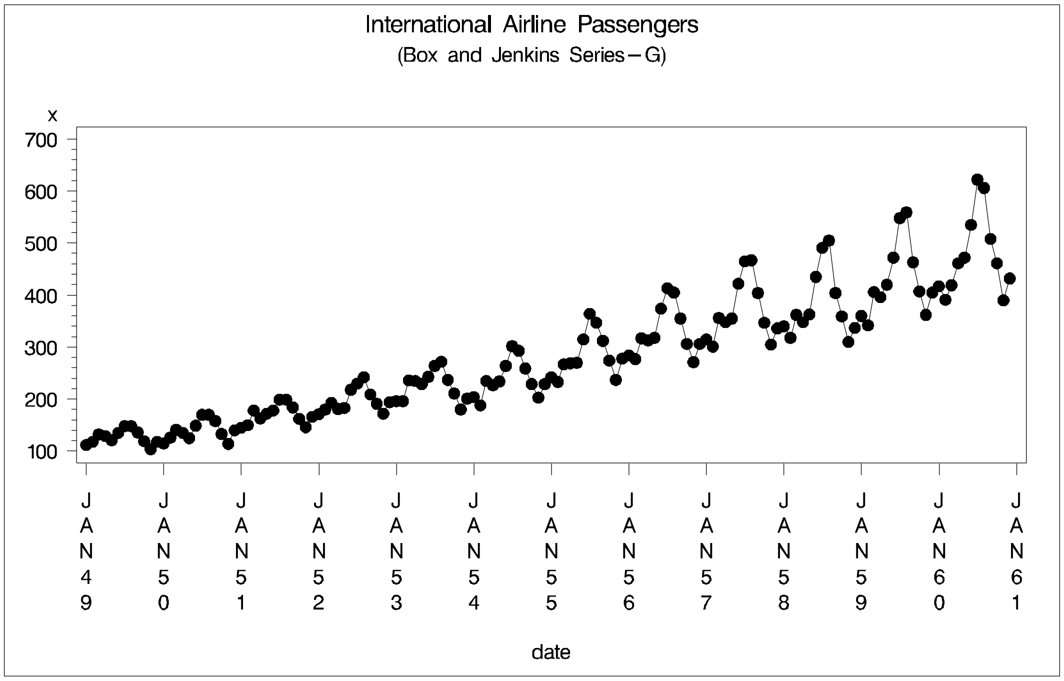
Example 11.2. Seasonal Model for the Airline Series

The airline passenger data, given as Series G in Box and Jenkins (1976), has been used in time series analysis literature as an example of a nonstationary seasonal time series. This example uses PROC ARIMA to fit the Airline model, ARIMA(0,1,1)$\times$(0,1,1)$_{12}$, to Box and Jenkins' "Series G."

The following statements read the data and log transform the series. The PROC GPLOT step plots the series, as shown in Output 11.2.1.

```
title1 'International Airline Passengers';
title2 '(Box and Jenkins Series-G)';
data seriesg;
   input x @@;
   xlog = log( x );
   date = intnx( 'month', '31dec1948'd, _n_ );
   format date monyy.;
   datalines;
112 118 132 129 121 135 148 148 136 119 104 118
115 126 141 135 125 149 170 170 158 133 114 140
145 150 178 163 172 178 199 199 184 162 146 166
171 180 193 181 183 218 230 242 209 191 172 194
196 196 236 235 229 243 264 272 237 211 180 201
204 188 235 227 234 264 302 293 259 229 203 229
242 233 267 269 270 315 364 347 312 274 237 278
284 277 317 313 318 374 413 405 355 306 271 306
315 301 356 348 355 422 465 467 404 347 305 336
340 318 362 348 363 435 491 505 404 359 310 337
360 342 406 396 420 472 548 559 463 407 362 405
417 391 419 461 472 535 622 606 508 461 390 432
;

symbol1 i=join  v=dot;
proc gplot data=seriesg;
   plot x * date = 1 / haxis= '1jan49'd to '1jan61'd by year;
run;
```

Output 11.2.1. Plot of Data



The following PROC ARIMA step fits an ARIMA(0,1,1)×(0,1,1)$_{12}$ model without a mean term to the logarithms of the airline passengers series. The model is forecast, and the results stored in the data set B.

```
proc arima data=seriesg;
```

```
   identify var=xlog(1,12) nlag=15;
   run;
   estimate q=(1)(12) noconstant method=uls;
   run;
   forecast out=b lead=24 id=date interval=month noprint;
quit;
```

The printed output from the IDENTIFY statement is shown in Output 11.2.2. The autocorrelation plots shown are for the twice differenced series $(1 - B)(1 - B^{12})X$. Note that the autocorrelation functions have the pattern characteristic of a first-order moving average process combined with a seasonal moving average process with lag 12.

Output 11.2.2. IDENTIFY Statement Output

```
                              The ARIMA Procedure

                            Name of Variable = xlog

             Period(s) of Differencing                        1,12
             Mean of Working Series                       0.000291
             Standard Deviation                           0.045673
             Number of Observations                            131
             Observation(s) eliminated by differencing          13


                                Autocorrelations

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0    0.0020860       1.00000   |                    |********************|
  1   -0.0007116       -.34112   |             *******|  .                 |
  2   0.00021913       0.10505   |                .   |** .                |
  3   -0.0004217       -.20214   |                ****|   .                |
  4   0.00004456       0.02136   |                .   |   .                |
  5   0.00011610       0.05565   |                .   |*  .                |
  6   0.00006426       0.03080   |                .   |*  .                |
  7   -0.0001159       -.05558   |                .  *|   .                |
  8   -1.5867E-6       -.00076   |                .   |   .                |
  9   0.00036791       0.17637   |                .   |****                |
 10   -0.0001593       -.07636   |                . **|   .                |
 11   0.00013431       0.06438   |                .   |*  .                |
 12   -0.0008065       -.38661   |            ********|   .                |
 13   0.00031624       0.15160   |               .    |*** .               |
 14   -0.0001202       -.05761   |               .   *|    .               |
 15   0.00031200       0.14957   |               .    |*** .               |

                         "." marks two standard errors
```

Output 11.2.2. (continued)

```
                       The ARIMA Procedure

                     Inverse Autocorrelations

Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  1        0.41027    |                 .  |********            |
  2        0.12711    |                 .  |***                 |
  3        0.10189    |                 .  |**.                 |
  4        0.01978    |                 .  |  .                 |
  5       -0.10310    |                 .**|  .                 |
  6       -0.11886    |                 .**|  .                 |
  7       -0.04088    |                 . *|  .                 |
  8       -0.05086    |                 . *|  .                 |
  9       -0.06022    |                 . *|  .                 |
 10        0.06460    |                 .  |* .                 |
 11        0.19907    |                 .  |****                |
 12        0.31709    |                 .  |******              |
 13        0.12434    |                 .  |**.                 |
 14        0.06583    |                 .  |* .                 |
 15        0.01515    |                 .  |  .                 |


                     Partial Autocorrelations

Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  1       -0.34112    |             *******|  .                 |
  2       -0.01281    |                 .  |  .                 |
  3       -0.19266    |                ****|  .                 |
  4       -0.12503    |                 ***|  .                 |
  5        0.03309    |                 .  |* .                 |
  6        0.03468    |                 .  |* .                 |
  7       -0.06019    |                 . *|  .                 |
  8       -0.02022    |                 .  |  .                 |
  9        0.22558    |                 .  |*****               |
 10        0.04307    |                 .  |* .                 |
 11        0.04659    |                 .  |* .                 |
 12       -0.33869    |             *******|  .                 |
 13       -0.10918    |                 .**|  .                 |
 14       -0.07684    |                 .**|  .                 |
 15       -0.02175    |                 .  |  .                 |
```

Output 11.2.2. (continued)

```
                           The ARIMA Procedure

                   Autocorrelation Check for White Noise

 To      Chi-         Pr >
Lag    Square   DF   ChiSq   ---------------Autocorrelations---------------

  6     23.27    6  0.0007   -0.341    0.105   -0.202    0.021    0.056    0.031
 12     51.47   12  <.0001   -0.056   -0.001    0.176   -0.076    0.064   -0.387
```

The results of the ESTIMATE statement are shown in Output 11.2.3.

Output 11.2.3. ESTIMATE Statement Output

```
                              The ARIMA Procedure

                     Unconditional Least Squares Estimation

                                  Standard                 Approx
      Parameter       Estimate       Error    t Value    Pr > |t|     Lag

      MA1,1            0.39594     0.08149       4.86      <.0001       1
      MA2,1            0.61331     0.07961       7.70      <.0001      12


                        Variance Estimate      0.001363
                        Std Error Estimate     0.036921
                        AIC                    -484.755
                        SBC                    -479.005
                        Number of Residuals         131


                           Correlations of Parameter
                                   Estimates

                         Parameter     MA1,1     MA2,1

                         MA1,1         1.000    -0.055
                         MA2,1        -0.055     1.000


                       Autocorrelation Check of Residuals

    To      Chi-         Pr >
   Lag    Square   DF   ChiSq   ---------------Autocorrelations---------------

     6      5.56    4  0.2349   0.022   0.024  -0.125  -0.129   0.057   0.065
    12      8.49   10  0.5816  -0.065  -0.042   0.102  -0.060   0.023   0.007
    18     13.23   16  0.6560   0.022   0.039   0.045  -0.162   0.035   0.001
    24     24.99   22  0.2978  -0.106  -0.104  -0.037  -0.027   0.219   0.040


                            Model for variable xlog

                       Period(s) of Differencing    1,12


                             Moving Average Factors

                         Factor 1:  1 - 0.39594 B**(1)
                         Factor 2:  1 - 0.61331 B**(12)
```

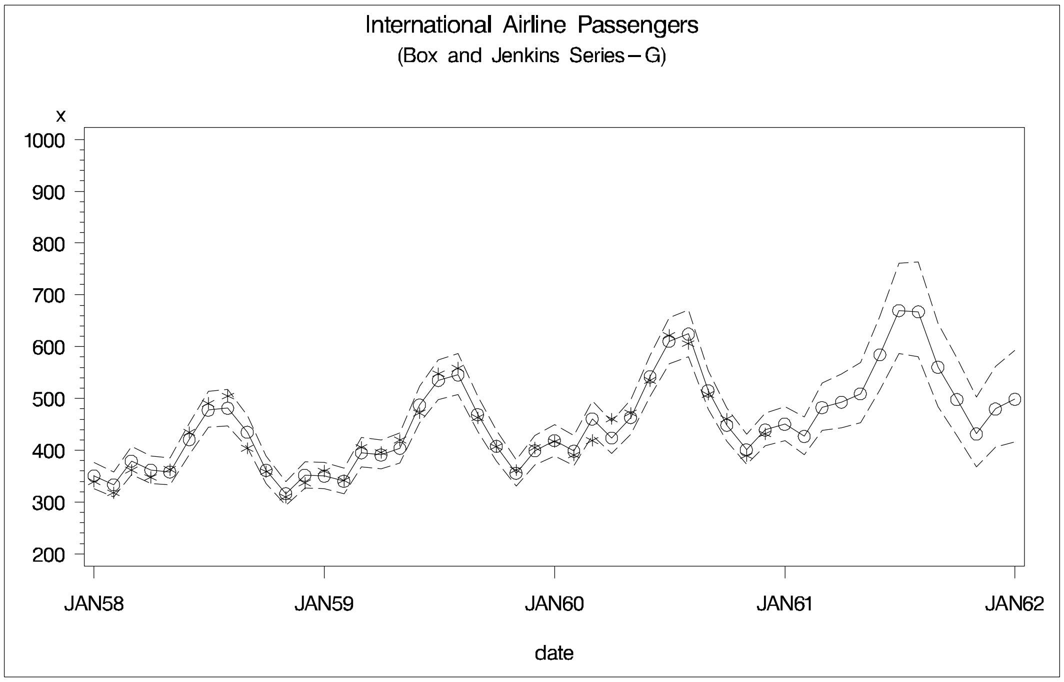
The following statements retransform the forecast values to get forecasts in the original scales. See the section "Forecasting Log Transformed Data" on page 429 for more information.

```
   data c;
      set b;
      x        = exp( xlog );
      forecast = exp( forecast + std*std/2 );
      l95      = exp( l95 );
      u95      = exp( u95 );
   run;
```

The forecasts and their confidence limits are plotted using the following PROC GPLOT step. The plot is shown in Output 11.2.4.

```
symbol1 i=none  v=star;
symbol2 i=join  v=circle;
symbol3 i=join  v=none l=3;
proc gplot data=c;
   where date >= '1jan58'd;
   plot x * date = 1 forecast * date = 2
        l95 * date = 3 u95 * date = 3 /
        overlay haxis= '1jan58'd to '1jan62'd by year;
run;
```

Output 11.2.4. Plot of the Forecast for the Original Series



Example 11.3. Model for Series J Data from Box and Jenkins

This example uses the Series J data from Box and Jenkins (1976). First the input series, *X*, is modeled with a univariate ARMA model. Next, the dependent series, *Y*, is cross correlated with the input series. Since a model has been fit to *X*, both *Y* and *X* are prewhitened by this model before the sample cross correlations are computed. Next, a transfer function model is fit with no structure on the noise term. The residuals from this model are identified by means of the PLOT option; then, the full model, transfer function and noise is fit to the data.

The following statements read Input Gas Rate and Output CO_2 from a gas furnace. (Data values are not shown. See "Series J" in Box and Jenkins (1976) for the values.)

```
title1 'Gas Furnace Data';
title2 '(Box and Jenkins, Series J)';
```

```
data seriesj;
   input x y @@;
   label x = 'Input Gas Rate'
         y = 'Output CO2';
datalines;
;
```

The following statements produce Output 11.3.1 through Output 11.3.5.

```
proc arima data=seriesj;

   /*--- Look at the input process -------------------*/
   identify var=x nlag=10;
   run;

   /*--- Fit a model for the input -------------------*/
   estimate p=3;
   run;

   /*--- Crosscorrelation of prewhitened series ------*/
   identify var=y crosscorr=(x) nlag=10;
   run;

   /*--- Fit transfer function - look at residuals ---*/
   estimate input=( 3 $ (1,2)/(1,2) x ) plot;
   run;

   /*--- Estimate full model ------------------------*/
   estimate p=2 input=( 3 $ (1,2)/(1) x );
   run;

quit;
```

The results of the first IDENTIFY statement for the input series X are shown in Output 11.3.1.

Output 11.3.1. IDENTIFY Statement Results for X

```
                                Gas Furnace Data
                          (Box and Jenkins, Series J)

                              The ARIMA Procedure

                              Name of Variable = x

                       Mean of Working Series    -0.05683
                       Standard Deviation        1.070952
                       Number of Observations         296


                                Autocorrelations

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0     1.146938       1.00000   |                    |********************|
  1     1.092430       0.95247   |                  . |******************* |
  2     0.956652       0.83409   |                .   |*****************   |
  3     0.782051       0.68186   |               .    |**************      |
  4     0.609291       0.53123   |               .    |***********         |
  5     0.467380       0.40750   |              .     |********            |
  6     0.364957       0.31820   |              .     |******              |
  7     0.298427       0.26019   |              .     |*****.              |
  8     0.260943       0.22751   |              .     |*****.              |
  9     0.244378       0.21307   |              .     |**** .              |
 10     0.238942       0.20833   |              .     |**** .              |

                         "." marks two standard errors


                            Inverse Autocorrelations

       Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

         1       -0.71090    |      **************| .                  |
         2        0.26217    |                  . |*****               |
         3       -0.13005    |                 ***| .                  |
         4        0.14777    |                  . |***                 |
         5       -0.06803    |                  .*| .                  |
         6       -0.01147    |                  . | .                  |
         7       -0.01649    |                  . | .                  |
         8        0.06108    |                  . |*.                  |
         9       -0.04490    |                  .*| .                  |
        10        0.01100    |                  . | .                  |
```

Output 11.3.1. (continued)

```
                              The ARIMA Procedure

                            Partial Autocorrelations

      Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

        1        0.95247    |                   . |******************* |
        2       -0.78796    |     ****************| .                  |
        3        0.33897    |                   . |*******             |
        4        0.12121    |                   . |**                  |
        5        0.05896    |                   . |*.                  |
        6       -0.11147    |                   **| .                  |
        7        0.04862    |                   . |*.                  |
        8        0.09945    |                   . |**                  |
        9        0.01587    |                   . | .                  |
       10       -0.06973    |                   .*| .                  |


                     Autocorrelation Check for White Noise

   To        Chi-           Pr >
  Lag      Square     DF    ChiSq    ---------------Autocorrelations---------------

    6      786.35      6   <.0001     0.952    0.834    0.682    0.531    0.408    0.318
```

The ESTIMATE statement results for the AR(3) model for the input series X are shown in Output 11.3.2.

Output 11.3.2. Estimates of the AR(3) Model for X

```
                              The ARIMA Procedure

                      Conditional Least Squares Estimation

                                   Standard                 Approx
      Parameter      Estimate         Error    t Value    Pr > |t|     Lag

      MU             -0.12280       0.10902      -1.13      0.2609       0
      AR1,1           1.97607       0.05499      35.94      <.0001       1
      AR1,2          -1.37499       0.09967     -13.80      <.0001       2
      AR1,3           0.34336       0.05502       6.24      <.0001       3


                        Constant Estimate      -0.00682
                        Variance Estimate      0.035797
                        Std Error Estimate       0.1892
                        AIC                    -141.667
                        SBC                    -126.906
                        Number of Residuals         296
                   * AIC and SBC do not include log determinant.


                      Correlations of Parameter Estimates

               Parameter         MU      AR1,1     AR1,2     AR1,3

               MU             1.000     -0.017     0.014    -0.016
               AR1,1         -0.017      1.000    -0.941     0.790
               AR1,2          0.014     -0.941     1.000    -0.941
               AR1,3         -0.016      0.790    -0.941     1.000


                       Autocorrelation Check of Residuals

    To      Chi-         Pr >
   Lag    Square   DF   ChiSq   ---------------Autocorrelations--------------

     6     10.30    3  0.0162  -0.042   0.068   0.056  -0.145  -0.009   0.059
    12     19.89    9  0.0186   0.014   0.002  -0.055   0.035   0.143  -0.079
    18     27.92   15  0.0221   0.099   0.043  -0.082   0.017   0.066  -0.052
    24     31.05   21  0.0729  -0.078   0.024   0.015   0.030   0.045   0.004
    30     34.58   27  0.1499  -0.007  -0.004   0.073  -0.038  -0.062   0.003
    36     38.84   33  0.2231   0.010   0.002   0.082   0.045   0.056  -0.023
    42     41.18   39  0.3753   0.002   0.033  -0.061  -0.003  -0.006  -0.043
    48     42.73   45  0.5687   0.018   0.051  -0.012   0.015  -0.027   0.020



                              The ARIMA Procedure

         Both variables have been prewhitened by the following filter:

                              Prewhitening Filter



                              The ARIMA Procedure

                   * AIC and SBC do not include log determinant.



                              The ARIMA Procedure

                   * AIC and SBC do not include log determinant.
```

Output 11.3.2. (continued)

```
                              The ARIMA Procedure

                              Model for variable x

                         Estimated Mean     -0.1228


                            Autoregressive Factors

         Factor 1:  1 - 1.97607 B**(1) + 1.37499 B**(2) - 0.34336 B**(3)
```

The IDENTIFY statement results for the dependent series Y cross correlated with the input series X is shown in Output 11.3.3. Since a model has been fit to X, both Y and X are prewhitened by this model before the sample cross correlations are computed.

Output 11.3.3. IDENTIFY Statement for Y Cross Correlated with X

```
                              The ARIMA Procedure

                            Partial Autocorrelations

     Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

       1        0.97076    |                   . |******************* |
       2       -0.80388    |    ****************| .                   |
       3        0.18833    |                   . |****                |
       4        0.25999    |                   . |*****               |
       5        0.05949    |                   . |*.                  |
       6       -0.06258    |                   .*| .                  |
       7       -0.01435    |                   . | .                  |
       8        0.05490    |                   . |*.                  |
       9        0.00545    |                   . | .                  |
      10        0.03141    |                   . |*.                  |


                      Autocorrelation Check for White Noise

  To       Chi-          Pr >
 Lag     Square    DF    ChiSq   ---------------Autocorrelations---------------

   6    1023.15     6   <.0001    0.971    0.896    0.793    0.680    0.574    0.485
```

Output 11.3.3. (continued)

```
                              The ARIMA Procedure

                             Correlation of y and x

                  Number of Observations                   296
                  Variance of transformed series y    0.131438
                  Variance of transformed series x    0.035357

                       Both series have been prewhitened.


                               Crosscorrelations

Lag    Covariance    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

-10     0.0015683        0.02301    |                  . | .                  |
 -9    0.00013502        0.00198    |                  . | .                  |
 -8    -0.0060480        -.08872    |                  **| .                  |
 -7    -0.0017624        -.02585    |                  .*| .                  |
 -6    -0.0080539        -.11814    |                  **| .                  |
 -5    -0.0000944        -.00138    |                  . | .                  |
 -4    -0.0012802        -.01878    |                  . | .                  |
 -3    -0.0031078        -.04559    |                  .*| .                  |
 -2    0.00065212        0.00957    |                  . | .                  |
 -1    -0.0019166        -.02811    |                  .*| .                  |
  0    -0.0003673        -.00539    |                  . | .                  |
  1     0.0038939        0.05712    |                  . |*.                  |
  2    -0.0016971        -.02489    |                  . | .                  |
  3     -0.019231        -.28210    |              ******| .                  |
  4     -0.022479        -.32974    |             *******| .                  |
  5     -0.030909        -.45341    |           *********| .                  |
  6     -0.018122        -.26583    |               *****| .                  |
  7     -0.011426        -.16761    |                 ***| .                  |
  8    -0.0017355        -.02546    |                  .*| .                  |
  9     0.0022590        0.03314    |                  . |*.                  |
 10    -0.0035152        -.05156    |                  .*| .                  |

                         "." marks two standard errors


                      Crosscorrelation Check Between Series

    To      Chi-          Pr >
   Lag    Square    DF    ChiSq    ---------------Crosscorrelations-------------

     5    117.75     6  <.0001    -0.005    0.057   -0.025   -0.282   -0.330   -0.453
```

Output 11.3.3. (continued)

```
                          The ARIMA Procedure

                * AIC and SBC do not include log determinant.


                          The ARIMA Procedure

        Both variables have been prewhitened by the following filter:

                          Prewhitening Filter


                         Autoregressive Factors

        Factor 1:  1 - 1.97607 B**(1) + 1.37499 B**(2) - 0.34336 B**(3)


                          The ARIMA Procedure

                * AIC and SBC do not include log determinant.


                          The ARIMA Procedure

                * AIC and SBC do not include log determinant.
```

The ESTIMATE statement results for the transfer function model with no structure on the noise term is shown in Output 11.3.4. The PLOT option prints the residual autocorrelation functions from this model.

Output 11.3.4. Estimates of the Transfer Function Model

```
                               The ARIMA Procedure

                  * AIC and SBC do not include log determinant.



                               The ARIMA Procedure

          Both variables have been prewhitened by the following filter:

                               Prewhitening Filter



                               The ARIMA Procedure

                      Conditional Least Squares Estimation

                             Standard                 Approx
  Parameter     Estimate        Error    t Value    Pr > |t|    Lag  Variable  Shift

  MU            53.32237      0.04932    1081.24      <.0001      0  y             0
  NUM1          -0.62868      0.25385      -2.48      0.0138      0  x             3
  NUM1,1         0.47258      0.62253       0.76      0.4484      1  x             3
  NUM1,2         0.73660      0.81006       0.91      0.3640      2  x             3
  DEN1,1         0.15411      0.90483       0.17      0.8649      1  x             3
  DEN1,2         0.27774      0.57345       0.48      0.6285      2  x             3


                         Constant Estimate      53.32237
                         Variance Estimate      0.704241
                         Std Error Estimate     0.839191
                         AIC                    729.7249
                         SBC                    751.7648
                         Number of Residuals         291
                  * AIC and SBC do not include log determinant.


                       Correlations of Parameter Estimates

 Variable                  y         x         x         x         x         x
 Parameter                MU      NUM1    NUM1,1    NUM1,2    DEN1,1    DEN1,2

 y              MU     1.000     0.013     0.002    -0.002     0.004    -0.006
 x            NUM1     0.013     1.000     0.755    -0.447     0.089    -0.065
 x          NUM1,1     0.002     0.755     1.000     0.121    -0.538     0.565
 x          NUM1,2    -0.002    -0.447     0.121     1.000    -0.892     0.870
 x          DEN1,1     0.004     0.089    -0.538    -0.892     1.000    -0.998
 x          DEN1,2    -0.006    -0.065     0.565     0.870    -0.998     1.000



                               The ARIMA Procedure

                  * AIC and SBC do not include log determinant.
```

Output 11.3.4. (continued)

```
                              The ARIMA Procedure

                       Autocorrelation Check of Residuals

    To      Chi-         Pr >
   Lag    Square   DF   ChiSq  ---------------Autocorrelations---------------

     6    496.45    6  <.0001   0.893   0.711   0.502   0.312   0.167   0.064
    12    498.58   12  <.0001  -0.003  -0.040  -0.054  -0.040  -0.022  -0.021
    18    539.38   18  <.0001  -0.045  -0.083  -0.131  -0.170  -0.196  -0.195
    24    561.87   24  <.0001  -0.163  -0.102  -0.026   0.047   0.106   0.142
    30    585.90   30  <.0001   0.158   0.156   0.131   0.081   0.013  -0.037
    36    592.42   36  <.0001  -0.048  -0.018   0.038   0.070   0.079   0.067
    42    593.44   42  <.0001   0.042   0.025   0.013   0.004   0.006   0.019
    48    601.94   48  <.0001   0.043   0.068   0.084   0.082   0.061   0.023
```

Output 11.3.4. (continued)

```
                              The ARIMA Procedure

                       Autocorrelation Plot of Residuals

Lag   Covariance   Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

  0     0.704241       1.00000   |                    |********************|
  1     0.628846       0.89294   |                  . |******************  |
  2     0.500490       0.71068   |                .   |**************      |
  3     0.353404       0.50182   |                .   |**********          |
  4     0.219895       0.31224   |               .    |******              |
  5     0.117330       0.16660   |               .    |*** .               |
  6     0.044967       0.06385   |               .    |*   .               |
  7   -0.0023551       -.00334   |               .    |    .               |
  8    -0.028030       -.03980   |               .   *|    .               |
  9    -0.037891       -.05380   |               .   *|    .               |
 10    -0.028378       -.04030   |               .   *|    .               |

                         "." marks two standard errors


                            Inverse Autocorrelations

       Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

         1       -0.57346    |         ***********| .                  |
         2        0.02264    |                  . | .                  |
         3        0.03631    |                  . |*.                  |
         4        0.03941    |                  . |*.                  |
         5       -0.01256    |                  . | .                  |
         6       -0.01618    |                  . | .                  |
         7        0.02680    |                  . |*.                  |
         8       -0.05895    |                  .*| .                  |
         9        0.07043    |                  . |*.                  |
        10       -0.02987    |                  .*| .                  |
```

Output 11.3.4. (continued)

```
                              The ARIMA Procedure

                            Partial Autocorrelations

     Lag    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

       1        0.89294    |                   . |******************  |
       2       -0.42765    |            *********| .                  |
       3       -0.13463    |                  ***| .                  |
       4        0.02199    |                   . | .                  |
       5        0.03891    |                   . |*.                  |
       6       -0.02219    |                   . | .                  |
       7       -0.02249    |                   . | .                  |
       8        0.01538    |                   . | .                  |
       9        0.00634    |                   . | .                  |
      10        0.07737    |                   . |**                  |


                 Crosscorrelation Check of Residuals with Input x

  To      Chi-         Pr >
 Lag    Square    DF   ChiSq   ---------------Crosscorrelations-------------

   5      0.48     2  0.7855  -0.009  -0.005   0.026   0.013  -0.017  -0.022
  11      0.93     8  0.9986  -0.006   0.008   0.022   0.023  -0.017  -0.013
  17      2.63    14  0.9996   0.012   0.035   0.037   0.039  -0.005  -0.040
  23     19.19    20  0.5092  -0.076  -0.108  -0.122  -0.122  -0.094  -0.041
  29     20.12    26  0.7857  -0.039  -0.013   0.010  -0.020  -0.031  -0.005
  35     24.22    32  0.8363  -0.022  -0.031  -0.074  -0.036   0.014   0.076
  41     30.66    38  0.7953   0.108   0.091   0.046   0.018   0.003   0.009
  47     31.65    44  0.9180   0.008  -0.011  -0.040  -0.030  -0.002   0.028


                               Model for variable y

                          Estimated Intercept    53.32237


                                  Input Number 1

                               Input Variable    x
                               Shift             3


                                Numerator Factors

                 Factor 1:  -0.6287 - 0.47258 B**(1) - 0.7366 B**(2)


                               Denominator Factors

                   Factor 1:  1 - 0.15411 B**(1) - 0.27774 B**(2)
```

The ESTIMATE statement results for the final transfer function model with AR(2) noise are shown in Output 11.3.5.

Output 11.3.5. Estimates of the Final Model

```
                                The ARIMA Procedure

                         Conditional Least Squares Estimation

                               Standard                 Approx
Parameter      Estimate           Error    t Value    Pr > |t|     Lag   Variable   Shift

MU             53.26307         0.11926     446.63      <.0001       0   y              0
AR1,1           1.53292         0.04754      32.25      <.0001       1   y              0
AR1,2          -0.63297         0.05006     -12.64      <.0001       2   y              0
NUM1           -0.53522         0.07482      -7.15      <.0001       0   x              3
NUM1,1          0.37602         0.10287       3.66      0.0003       1   x              3
NUM1,2          0.51894         0.10783       4.81      <.0001       2   x              3
DEN1,1          0.54842         0.03822      14.35      <.0001       1   x              3


                         Constant Estimate      5.329371
                         Variance Estimate      0.058828
                         Std Error Estimate     0.242544
                         AIC                    8.292811
                         SBC                    34.00607
                         Number of Residuals         291
                  * AIC and SBC do not include log determinant.


                          Correlations of Parameter Estimates

         Variable                   y           y           y           x
         Parameter                 MU       AR1,1       AR1,2        NUM1

         y               MU     1.000      -0.063       0.047      -0.008
         y            AR1,1    -0.063       1.000      -0.927      -0.003
         y            AR1,2     0.047      -0.927       1.000       0.023
         x             NUM1    -0.008      -0.003       0.023       1.000
         x           NUM1,1    -0.016       0.007      -0.005       0.713
         x           NUM1,2     0.017      -0.002       0.005      -0.178
         x           DEN1,1    -0.049       0.015      -0.022      -0.013

                          Correlations of Parameter Estimates

              Variable                   x           x           x
              Parameter             NUM1,1      NUM1,2      DEN1,1

              y               MU    -0.016       0.017      -0.049
              y            AR1,1     0.007      -0.002       0.015
              y            AR1,2    -0.005       0.005      -0.022
              x             NUM1     0.713      -0.178      -0.013
              x           NUM1,1     1.000      -0.467      -0.039
              x           NUM1,2    -0.467       1.000      -0.720
              x           DEN1,1    -0.039      -0.720       1.000
```

Output 11.3.5. (continued)

```
                              The ARIMA Procedure

                      Autocorrelation Check of Residuals

 To        Chi-          Pr >
Lag      Square    DF   ChiSq   --------------Autocorrelations--------------

  6        8.61     4  0.0717    0.024    0.055   -0.073   -0.054   -0.054    0.119
 12       15.43    10  0.1172    0.032    0.028   -0.081    0.047    0.022    0.107
 18       21.13    16  0.1734   -0.038    0.052   -0.093   -0.013   -0.073   -0.005
 24       27.52    22  0.1922   -0.118   -0.002   -0.007    0.076    0.024   -0.004
 30       36.94    28  0.1202    0.034   -0.021    0.020    0.094   -0.118    0.065
 36       44.26    34  0.1119   -0.025   -0.057    0.113    0.022    0.030    0.065
 42       45.62    40  0.2500   -0.017   -0.036   -0.029   -0.013   -0.033    0.017
 48       48.60    46  0.3689    0.024    0.069    0.024    0.017    0.022   -0.044


                 Crosscorrelation Check of Residuals with Input x

 To        Chi-          Pr >
Lag      Square    DF   ChiSq   --------------Crosscorrelations-------------

  5        0.93     3  0.8191    0.008    0.004    0.010    0.008   -0.045    0.030
 11        6.60     9  0.6784    0.075   -0.024   -0.019   -0.026   -0.111    0.013
 17       13.86    15  0.5365    0.050    0.043    0.014    0.014   -0.141   -0.028
 23       18.55    21  0.6142   -0.074   -0.078    0.023   -0.016    0.021    0.060
 29       27.99    27  0.4113   -0.071   -0.001    0.038   -0.156    0.031    0.035
 35       35.18    33  0.3654   -0.014    0.015   -0.039    0.028    0.046    0.142
 41       37.15    39  0.5544    0.031   -0.029   -0.070   -0.006    0.012   -0.004
 47       42.42    45  0.5818    0.036   -0.038   -0.053    0.107    0.029    0.021
```

Output 11.3.5. (continued)

```
                              The ARIMA Procedure

                             Model for variable y

                       Estimated Intercept    53.26307


                            Autoregressive Factors

                Factor 1:  1 - 1.53292 B**(1) + 0.63297 B**(2)


                                Input Number 1

                              Input Variable    x
                              Shift             3


                              Numerator Factors

             Factor 1:  -0.5352 - 0.37602 B**(1) - 0.51894 B**(2)


                             Denominator Factors

                         Factor 1:  1 - 0.54842 B**(1)
```

Example 11.4. An Intervention Model for Ozone Data

This example fits an intervention model to ozone data as suggested by Box and Tiao (1975). Notice that since the response variable, OZONE, is differenced, the innovation, X1, must also be differenced to generate a step function change in the response. If X1 had not been differenced, the change in the response caused by X1 would be a (seasonal) ramp and not a step function. Notice that the final model for the differenced data is a multiple regression model with a moving-average structure assumed for the residuals.

The model is fit by maximum likelihood. The seasonal moving-average parameter and its standard error are fairly sensitive to which method is chosen to fit the model, in agreement with the observations of Davidson (1981) and Ansley and Newbold (1980); thus, fitting the model by the unconditional or conditional least squares methods produce somewhat different estimates for these parameters.

Some missing values are appended to the end of the input data to generate additional values for the independent variables. Since the independent variables are not modeled, values for them must be available for any times at which predicted values are desired. In this case, predicted values are requested for 12 periods beyond the end of the data. Thus, values for X1, WINTER, and SUMMER must be given for 12 periods ahead.

The following statements read in the data and compute dummy variables for use as intervention inputs:

```
title1 'Intervention Data for Ozone Concentration';
title2 '(Box and Tiao, JASA 1975 P.70)';

data air;
   input ozone @@;
   label ozone  = 'Ozone Concentration'
         x1     = 'Intervention for post 1960 period'
         summer = 'Summer Months Intervention'
         winter = 'Winter Months Intervention';
   date = intnx( 'month', '31dec1954'd, _n_ );
   format date monyy.;
   month = month( date );
   year = year( date );
   x1 = year >= 1960;
   summer = ( 5 < month < 11 ) * ( year > 1965 );
   winter = ( year > 1965 ) - summer;
datalines;
2.7  2.0  3.6  5.0  6.5  6.1  5.9  5.0  6.4  7.4  8.2  3.9
4.1  4.5  5.5  3.8  4.8  5.6  6.3  5.9  8.7  5.3  5.7  5.7
3.0  3.4  4.9  4.5  4.0  5.7  6.3  7.1  8.0  5.2  5.0  4.7
3.7  3.1  2.5  4.0  4.1  4.6  4.4  4.2  5.1  4.6  4.4  4.0
2.9  2.4  4.7  5.1  4.0  7.5  7.7  6.3  5.3  5.7  4.8  2.7
1.7  2.0  3.4  4.0  4.3  5.0  5.5  5.0  5.4  3.8  2.4  2.0
2.2  2.5  2.6  3.3  2.9  4.3  4.2  4.2  3.9  3.9  2.5  2.2
2.4  1.9  2.1  4.5  3.3  3.4  4.1  5.7  4.8  5.0  2.8  2.9
1.7  3.2  2.7  3.0  3.4  3.8  5.0  4.8  4.9  3.5  2.5  2.4
```

```
1.6  2.3  2.5  3.1  3.5  4.5  5.7  5.0  4.6  4.8  2.1  1.4
2.1  2.9  2.7  4.2  3.9  4.1  4.6  5.8  4.4  6.1  3.5  1.9
1.8  1.9  3.7  4.4  3.8  5.6  5.7  5.1  5.6  4.8  2.5  1.5
1.8  2.5  2.6  1.8  3.7  3.7  4.9  5.1  3.7  5.4  3.0  1.8
2.1  2.6  2.8  3.2  3.5  3.5  4.9  4.2  4.7  3.7  3.2  1.8
2.0  1.7  2.8  3.2  4.4  3.4  3.9  5.5  3.8  3.2  2.3  2.2
1.3  2.3  2.7  3.3  3.7  3.0  3.8  4.7  4.6  2.9  1.7  1.3
1.8  2.0  2.2  3.0  2.4  3.5  3.5  3.3  2.7  2.5  1.6  1.2
1.5  2.0  3.1  3.0  3.5  3.4  4.0  3.8  3.1  2.1  1.6  1.3
 .    .    .    .    .    .    .    .    .    .    .    .
;
```

The following statements produce Output 11.4.1 and Output 11.4.2:

```
proc arima data=air;

   /*--- Identify and seasonally difference ozone series ---*/
   identify var=ozone(12) crosscorr=( x1(12) summer winter ) noprint;

   /*--- Fit a multiple regression with a seasonal MA model ---*/
   /*---     by the maximum likelihood method ---*/
   estimate q=(1)(12) input=( x1 summer winter )
            noconstant method=ml itprint;

   /*--- Forecast ---*/
   forecast  lead=12 id=date interval=month;

run;
```

The ESTIMATE statement results are shown in Output 11.4.1.

Output 11.4.1. Parameter Estimates

```
                  Intervention Data for Ozone Concentration
                        (Box and Tiao, JASA 1975 P.70)

                             The ARIMA Procedure

                           Initial Moving Average
                                  Estimates

                                         Estimate

                             1           -0.29241


                           Initial Moving Average
                                  Estimates

                                         Estimate

                            12            0.40740


                     White Noise Variance Est    0.944969
```

Output 11.4.1. (continued)

```
                              The ARIMA Procedure

                      Conditional Least Squares Estimation

Iteration       SSE     MA1,1     MA2,1      NUM1      NUM2      NUM3    Lambda

        0    154.53  -0.29241   0.40740  -1.13490  -0.11731   0.05581   0.00001
        1    146.20  -0.29256   0.59844  -1.20292  -0.29784  -0.11572      1E-6
        2    145.88  -0.30071   0.59239  -1.26173  -0.26252  -0.08247      1E-7
        3    145.88  -0.29976   0.59242  -1.26246  -0.26150  -0.08197      1E-8
        4    145.88  -0.29983   0.59234  -1.26243  -0.26154  -0.08196      1E-9

                               Conditional Least
                               Squares Estimation

                              Iteration    R Crit

                                      0         1
                                      1  0.230552
                                      2  0.046601
                                      3  0.001345
                                      4  0.000125


                          Maximum Likelihood Estimation

 Iter    Loglike    MA1,1    MA2,1     NUM1     NUM2     NUM3   Lambda   R Crit

    0 -249.07778 -0.29983  0.59234 -1.26243 -0.26154 -0.08196  0.00001        1
    1 -245.89135 -0.26830  0.76634 -1.34490 -0.23984 -0.07578     1E-6 0.169445
    2 -245.88484 -0.26653  0.76623 -1.33046 -0.23939 -0.08025     1E-7 0.008044
    3 -245.88482 -0.26689  0.76661 -1.33070 -0.23936 -0.08020     1E-8 0.000603
    4 -245.88481 -0.26684  0.76665 -1.33062 -0.23936 -0.08021     1E-9 0.000073


                     ARIMA Estimation Optimization Summary

Estimation Method                                             Maximum Likelihood
Parameters Estimated                                                           5
Termination Criteria                          Maximum Relative Change in Estimates
Iteration Stopping Value                                                   0.001
Criteria Value                                                          0.000195
Alternate Criteria                           Relative Change in Objective Function
Alternate Criteria Value                                                1.247E-8
Maximum Absolute Value of Gradient                                       0.00712
R-Square Change from Last Iteration                                     0.000073
Objective Function                                         Log Gaussian Likelihood
Objective Function Value                                                -245.885
Marquardt's Lambda Coefficient                                              1E-9
Numerical Derivative Perturbation Delta                                    0.001
Iterations                                                                     4
```

Output 11.4.1. (continued)

```
                              The ARIMA Procedure

                         Maximum Likelihood Estimation

                              Standard                 Approx
Parameter      Estimate          Error    t Value    Pr > |t|     Lag   Variable   Shift

MA1,1          -0.26684        0.06710      -3.98      <.0001       1   ozone          0
MA2,1           0.76665        0.05973      12.83      <.0001      12   ozone          0
NUM1           -1.33062        0.19236      -6.92      <.0001       0   x1             0
NUM2           -0.23936        0.05952      -4.02      <.0001       0   summer         0
NUM3           -0.08021        0.04978      -1.61      0.1071       0   winter         0


                        Variance Estimate       0.634506
                        Std Error Estimate      0.796559
                        AIC                     501.7696
                        SBC                     518.3602
                        Number of Residuals          204


                       Correlations of Parameter Estimates

    Variable                 ozone      ozone         x1     summer     winter
    Parameter                MA1,1      MA2,1       NUM1       NUM2       NUM3

    ozone        MA1,1       1.000      0.090     -0.039      0.062     -0.034
    ozone        MA2,1       0.090      1.000     -0.169      0.211      0.022
    x1            NUM1      -0.039     -0.169      1.000     -0.124     -0.107
    summer        NUM2       0.062      0.211     -0.124      1.000      0.097
    winter        NUM3      -0.034      0.022     -0.107      0.097      1.000


                        Autocorrelation Check of Residuals

  To       Chi-          Pr >
 Lag     Square    DF   ChiSq   ---------------Autocorrelations--------------

   6       7.47     4  0.1132    0.017    0.054    0.043    0.101   -0.022    0.140
  12      10.21    10  0.4220   -0.024   -0.059   -0.047    0.014    0.032    0.072
  18      14.53    16  0.5593    0.054    0.006   -0.110    0.028   -0.042    0.043
  24      19.99    22  0.5834    0.003   -0.074   -0.074    0.098   -0.038    0.043
  30      27.00    28  0.5180   -0.072   -0.035    0.023   -0.028   -0.107    0.100
  36      32.65    34  0.5336    0.022   -0.099   -0.006    0.087   -0.046    0.053
```

Output 11.4.1. (continued)

```
                              The ARIMA Procedure

                           Model for variable ozone

                        Period(s) of Differencing    12


                             Moving Average Factors

                         Factor 1:  1 + 0.26684 B**(1)
                         Factor 2:  1 - 0.76665 B**(12)


                                Input Number 1

                     Input Variable                   x1
                     Period(s) of Differencing        12
                     Overall Regression Factor  -1.33062


                                Input Number 2

                     Input Variable               summer
                     Overall Regression Factor  -0.23936


                                Input Number 3

                     Input Variable               winter
                     Overall Regression Factor  -0.08021
```

The FORECAST statement results are shown in Output 11.4.2.

Output 11.4.2. Forecasts

```
                              The ARIMA Procedure

                          Forecasts for variable ozone

           Obs       Forecast    Std Error       95% Confidence Limits

           217         1.4205       0.7966       -0.1407         2.9817
           218         1.8446       0.8244        0.2287         3.4604
           219         2.4567       0.8244        0.8408         4.0725
           220         2.8590       0.8244        1.2431         4.4748
           221         3.1501       0.8244        1.5342         4.7659
           222         2.7211       0.8244        1.1053         4.3370
           223         3.3147       0.8244        1.6989         4.9306
           224         3.4787       0.8244        1.8629         5.0946
           225         2.9405       0.8244        1.3247         4.5564
           226         2.3587       0.8244        0.7429         3.9746
           227         1.8588       0.8244        0.2429         3.4746
           228         1.2898       0.8244       -0.3260         2.9057
```

Example 11.5. Using Diagnostics to Identify ARIMA models

Fitting ARIMA models is as much an art as it is a science. The ARIMA procedure has diagnostic options to help tentatively identify the orders of both stationary and nonstationary ARIMA processes.

Consider the Series A in Box et al. (1994), which consists of 197 concentration readings taken every two hours from a chemical process. Let Series A be a data set containing these readings in a variable named X. The following SAS statements use the SCAN option of the IDENTIFY statement to generate Output 11.5.1 and Output 11.5.2. See "The SCAN Method" for details of the SCAN method.

```
proc arima data=SeriesA;
   identify var=x scan;
run;
```

Output 11.5.1. Example of SCAN Tables

```
                SERIES A: Chemical Process Concentration Readings

                               The ARIMA Procedure

                      Squared Canonical Correlation Estimates

         Lags      MA 0      MA 1      MA 2      MA 3      MA 4      MA 5

         AR 0    0.3263    0.2479    0.1654    0.1387    0.1183    0.1417
         AR 1    0.0643    0.0012    0.0028    <.0001    0.0051    0.0002
         AR 2    0.0061    0.0027    0.0021    0.0011    0.0017    0.0079
         AR 3    0.0072    <.0001    0.0007    0.0005    0.0019    0.0021
         AR 4    0.0049    0.0010    0.0014    0.0014    0.0039    0.0145
         AR 5    0.0202    0.0009    0.0016    <.0001    0.0126    0.0001


                       SCAN Chi-Square[1] Probability Values

         Lags      MA 0      MA 1      MA 2      MA 3      MA 4      MA 5

         AR 0    <.0001    <.0001    <.0001    0.0007    0.0037    0.0024
         AR 1    0.0003    0.6649    0.5194    0.9235    0.3993    0.8528
         AR 2    0.2754    0.5106    0.5860    0.7346    0.6782    0.2766
         AR 3    0.2349    0.9812    0.7667    0.7861    0.6810    0.6546
         AR 4    0.3297    0.7154    0.7113    0.6995    0.5807    0.2205
         AR 5    0.0477    0.7254    0.6652    0.9576    0.2660    0.9168
```

In Output 11.5.1, there is one (maximal) rectangular region in which all the elements are insignificant with 95% confidence. This region has a vertex at (1,1). Output 11.5.2 gives recommendations based on the significance level specified by the ALPHA=*siglevel* option.

Output 11.5.2. Example of SCAN Option Tentative Order Selection

```
                              The ARIMA Procedure

                                  ARMA(p+d,q)
                                   Tentative
                                     Order
                                   Selection
                                     Tests

                                  ----SCAN---
                                  p+d       q

                                    1       1

                            (5% Significance Level)
```

Another order identification diagnostic is the extended sample autocorrelation function or ESACF method. See "The ESACF Method" for details of the ESACF method.

The following statements generate Output 11.5.3 and Output 11.5.4.

```
proc arima data=SeriesA;
   identify var=x esacf;
run;
```

Output 11.5.3. Example of ESACF Tables

```
                              The ARIMA Procedure

                    Extended Sample Autocorrelation Function

         Lags      MA 0      MA 1      MA 2      MA 3      MA 4      MA 5

         AR 0    0.5702    0.4951    0.3980    0.3557    0.3269    0.3498
         AR 1   -0.3907    0.0425   -0.0605   -0.0083   -0.0651   -0.0127
         AR 2   -0.2859   -0.2699   -0.0449    0.0089   -0.0509   -0.0140
         AR 3   -0.5030   -0.0106    0.0946   -0.0137   -0.0148   -0.0302
         AR 4   -0.4785   -0.0176    0.0827   -0.0244   -0.0149   -0.0421
         AR 5   -0.3878   -0.4101   -0.1651    0.0103   -0.1741   -0.0231


                            ESACF Probability Values

         Lags      MA 0      MA 1      MA 2      MA 3      MA 4      MA 5

         AR 0    <.0001    <.0001    0.0001    0.0014    0.0053    0.0041
         AR 1    <.0001    0.5974    0.4622    0.9198    0.4292    0.8768
         AR 2    <.0001    0.0002    0.6106    0.9182    0.5683    0.8592
         AR 3    <.0001    0.9022    0.2400    0.8713    0.8930    0.7372
         AR 4    <.0001    0.8380    0.3180    0.7737    0.8913    0.6213
         AR 5    <.0001    <.0001    0.0765    0.9142    0.1038    0.8103
```

In Output 11.5.3, there are three right-triangular regions in which all elements are insignificant at the 5% level. The triangles have vertices (1,1), (3,1), and (4,1). Since the triangle at (1,1) covers more insignificant terms, it is recommended first. Similarly, the remaining recommendations are ordered by the number of insignificant terms contained in the triangle. Output 11.5.4 gives recommendations based on the significance level specified by the ALPHA=*siglevel* option.

Output 11.5.4. Example of ESACF Option Tentative Order Selection

```
                              The ARIMA Procedure

                                  ARMA(p+d,q)
                                   Tentative
                                     Order
                                   Selection
                                     Tests

                                  ---ESACF---
                                  p+d       q

                                    1       1
                                    3       1
                                    4       1

                            (5% Significance Level)
```

If you also specify the SCAN option in the same IDENTIFY statement, the two recommendations are printed side by side.

```
proc arima data=SeriesA;
   identify var=x scan esacf;
run;
```

Output 11.5.5. Example of SCAN and ESACF Option Combined

```
                              The ARIMA Procedure

                              ARMA(p+d,q) Tentative
                              Order Selection Tests

                             ---SCAN--    --ESACF--
                             p+d     q    p+d     q

                               1     1      1     1
                                            3     1
                                            4     1

                            (5% Significance Level)
```

From above, the autoregressive and moving average orders are tentatively identified by both SCAN and ESACF tables to be $(p + d, q)$=(1,1). Because both the SCAN and ESACF indicate a $p + d$ term of 1, a unit root test should be used to determine whether this autoregressive term is a unit root. Since a moving average term appears to be present, a large autoregressive term is appropriate for the Augmented Dickey-Fuller test for a unit root.

Submitting the following code generates Output 11.5.6.

```
proc arima data=SeriesA;
   identify var=x stationarity=(adf=(5,6,7,8));
run;
```

Output 11.5.6. Example of STATIONARITY Option Output

```
                              The ARIMA Procedure

                     Augmented Dickey-Fuller Unit Root Tests

Type           Lags        Rho    Pr < Rho        Tau    Pr < Tau          F    Pr > F

Zero Mean         5     0.0403      0.6913       0.42      0.8024
                  6     0.0479      0.6931       0.63      0.8508
                  7     0.0376      0.6907       0.49      0.8200
                  8     0.0354      0.6901       0.48      0.8175
Single Mean       5   -18.4550      0.0150      -2.67      0.0821       3.67    0.1367
                  6   -10.8939      0.1043      -2.02      0.2767       2.27    0.4931
                  7   -10.9224      0.1035      -1.93      0.3172       2.00    0.5605
                  8   -10.2992      0.1208      -1.83      0.3650       1.81    0.6108
Trend             5   -18.4360      0.0871      -2.66      0.2561       3.54    0.4703
                  6   -10.8436      0.3710      -2.01      0.5939       2.04    0.7694
                  7   -10.7427      0.3773      -1.90      0.6519       1.91    0.7956
                  8   -10.0370      0.4236      -1.79      0.7081       1.74    0.8293
```

The preceding test results show that a unit root is very likely and that the series should be differenced. Based on this test and the previous results, an ARIMA(0,1,1) would be a good choice for a tentative model for Series A.

Using the recommendation that the series be differenced, the following statements generate Output 11.5.7.

```
proc arima data=SeriesA;
   identify var=x(1) minic;
run;
```

Output 11.5.7. Example of MINIC Table

```
                              The ARIMA Procedure

                          Minimum Information Criterion

         Lags      MA 0       MA 1       MA 2       MA 3       MA 4       MA 5

         AR 0  -2.05761    -2.3497  -2.32358  -2.31298  -2.30967  -2.28528
         AR 1  -2.23291  -2.32345  -2.29665  -2.28644  -2.28356  -2.26011
         AR 2  -2.23947  -2.30313  -2.28084  -2.26065  -2.25685  -2.23458
         AR 3  -2.25092  -2.28088  -2.25567  -2.23455  -2.22997  -2.20769
         AR 4  -2.25934    -2.2778  -2.25363  -2.22983  -2.20312  -2.19531
         AR 5   -2.2751  -2.26805  -2.24249  -2.21789  -2.19667  -2.17426
```

The error series is estimated using an AR(7) model, and the minimum of this MINIC table is $BIC(0, 1)$. This diagnostic confirms the previous result indicating that an ARIMA(0,1,1) is a tentative model for Series A.

If you also specify the SCAN or MINIC option in the same IDENTIFY statement, the BIC associated with the SCAN table and ESACF table recommendations are listed.

```
proc arima data=SeriesA;
   identify var=x(1) minic scan esacf;
run;
```

Output 11.5.8. Example of SCAN, ESACF, MINIC Options Combined

```
                          The ARIMA Procedure

               ARMA(p+d,q) Tentative Order Selection Tests

             ---------SCAN--------    --------ESACF--------
             p+d     q         BIC    p+d     q         BIC

               0     1     -2.3497      0     1     -2.3497
                                        1     1    -2.32345

                        (5% Significance Level)
```

Example 11.6. Detection of Level Changes in the Nile River Data

This example is discussed in de Jong and Penzer (1998). The data consist of readings of the annual flow volume of the Nile River at Aswan from 1871 to 1970. These data have also been studied by Cobb (1978). These studies indicate that levels in the years 1877 and 1913 are strong candidates for additive outliers, and that there was a shift in the flow levels starting from the year 1899. This shift in 1899 is attributed partly to the weather changes and partly to the start of construction work for a new dam at Aswan.

```
data nile;
   input level @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
   datalines;
1120  1160  963  1210  1160  1160  813  1230   1370  1140
995   935   1110 994   1020  960   1180 799    958   1140
1100  1210  1150 1250  1260  1220  1030 1100   774   840
874   694   940  833   701   916   692  1020   1050  969
831   726   456  824   702   1120  1100 832    764   821
768   845   864  862   698   845   744  796    1040  759
781   865   845  944   984   897   822  1010   771   676
649   846   812  742   801   1040  860  874    848   890
744   749   838  1050  918   986   797  923    975   815
1020  906   901  1170  912   746   919  718    714   740
;
run;
```

You can start the modeling process with the ARIMA(0, 1, 1) model, an ARIMA model close to the Structural model suggested in de Jong and Penzer (1998), and examine the parameter estimates, the residual autocorrelations, and the outliers.

```
proc arima data=nile;
   identify var= level(1) noprint;
   estimate q = 1 noint method= ml plot;
   outlier maxnum= 5 id=year;
run;
```

A portion of the estimation and the outlier detection output is shown in Output 11.6.1.

Output 11.6.1. ARIMA(0, 1, 1) Model

```
                                  The ARIMA Procedure

                               Outlier Detection Summary

                           Maximum number searched           5
                           Number found                      5
                           Significance used              0.05


                                    Outlier Details
                                                                           Approx
                                                                   Chi-     Prob>
 Obs    Time ID         Type                      Estimate       Square     ChiSq

  29           1899    Shift                    -315.75346        13.13    0.0003
  43           1913    Additive                 -403.97105        11.83    0.0006
   7           1877    Additive                 -335.49351         7.69    0.0055
  94           1964    Additive                  305.03568         6.16    0.0131
  18           1888    Additive                 -287.81484         6.00    0.0143
```

Note that the first three outliers detected are indeed the ones discussed earlier. You can include the shock signatures corresponding to these three outliers in the Nile data set.

```
data nile;
   set nile;
      if year = '1jan1877'd then AO1877 = 1.0;
      else AO1877 = 0.0;
      if year = '1jan1913'd then AO1913 = 1.0;
      else AO1913 = 0.0;
      if year >= '1jan1899'd then LS1899 = 1.0;
      else LS1899 = 0.0;
run;
```

Now you can refine the earlier model by including these outliers. After examining the parameter estimates and residuals (not shown) of the ARIMA(0, 1, 1) model with these regressors, the following stationary MA1 model (with regressors) appears to fit the data well.

```
proc arima data=nile;
   identify var= level crosscorr= ( AO1877 AO1913 LS1899 ) noprint;
   estimate q = 1
   input= (AO1877 AO1913 LS1899) method= ml plot;
   outlier maxnum= 5 alpha= 0.01 id=year;
run;
```

The relevant outlier detection process output is shown in Output 11.6.2. No outliers, at significance level 0.01, were detected.

Output 11.6.2. MA1 Model with Outliers

```
                         The ARIMA Procedure

                      Outlier Detection Summary

                 Maximum number searched           5
                 Number found                      0
                 Significance used              0.01
```

Example 11.7. Iterative Outlier Detection

This example illustrates the iterative nature of the outlier detection process. This is done using a simple test example where an additive outlier at observation number 50 and a level shift at observation number 100 are artificially introduced in the International Airline Passenger data used in an earlier example (See Example 11.2). The following Data step shows the modifications introduced in the data set.

```
data airline;
   set sashelp.air;
   logair = log(air);
   if _n_ = 50 then logair = logair - 0.25;
   if _n_ >= 100 then logair = logair + 0.5;
run;
```

In Example 11.2 the Airline model, ARIMA$(0, 1, 1) \times (0, 1, 1)_{12}$, was seen to be a good fit to the unmodified log-transformed airline passenger series. The preliminary identification steps (not shown) again suggest the Airline model as a suitable initial model for the modified data.

```
proc arima data=airline;
   identify var=logair( 1, 12 )  noprint;
   estimate q = (1)(12) noint method= ml;
   outlier maxnum= 3 alpha= 0.01;
run;
```

A portion of the estimation and outlier detection output is shown in Output 11.7.1.

Output 11.7.1. Initial Model

```
                                The ARIMA Procedure

                             Outlier Detection Summary

                        Maximum number searched           3
                        Number found                      3
                        Significance used              0.01


                                  Outlier Details
                                                                  Approx
                                                          Chi-     Prob>
           Obs     Type                  Estimate       Square     ChiSq

           100     Shift                  0.49325       199.36    <.0001
            50     Additive              -0.27508       104.78    <.0001
           135     Additive              -0.10488        13.08    0.0003
```

Clearly the level shift at observation number 100 and the additive outlier at observation number 50 are the dominant outliers. Moreover, the corresponding regression coefficients seem to correctly estimate the size and sign of the change. You can augment the airline data with these two regressors.

```
   data airline;
      set airline;
      if _n_ = 50 then AO = 1;
      else AO = 0.0;
      if _n_ >= 100 then LS  = 1;
      else LS = 0.0;
   run;
```

You can now refine the previous model by including these regressors. Note that the differencing order of the dependent series is matched to the differencing orders of the outlier regressors to get the correct “effective” outlier signatures.

```
proc arima data=airline;
   identify var= logair(1, 12)
      crosscorr= ( AO(1, 12) LS(1, 12) ) noprint;
   estimate q = (1)(12) noint
      input= (AO LS) method= ml plot;
   outlier maxnum= 3 alpha= 0.01;
run;
```

The estimation and outlier detection results are shown in Output 11.7.2.

Output 11.7.2. Airline Model with Outliers

```
                              The ARIMA Procedure

                           Outlier Detection Summary

                      Maximum number searched          3
                      Number found                     3
                      Significance used             0.01


                                Outlier Details
                                                                    Approx
                                                           Chi-      Prob>
            Obs     Type                   Estimate      Square      ChiSq

            135     Additive               -0.10310       12.63     0.0004
             62     Additive               -0.08872       12.33     0.0004
             29     Additive                0.08686       11.66     0.0006
```

The output shows that a few outliers still remain to be accounted for and that the model could be refined further.

Example 11.8. Illustration of ODS Graphics (Experimental)

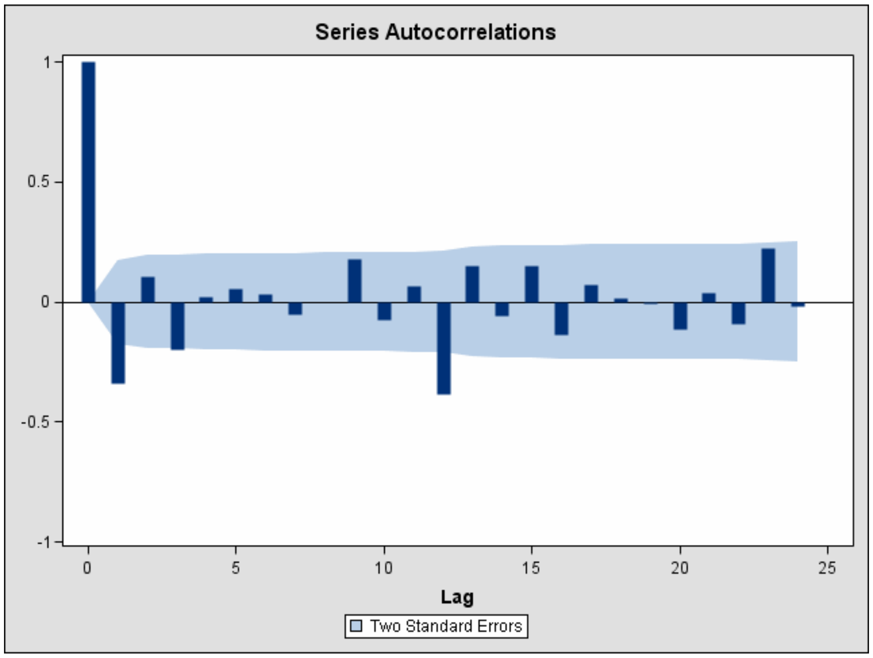
This example illustrates the use of experimental ODS graphics. The graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the ARIMA procedure, see the "ODS Graphics" section on page 443.

The following code shows how you can use the ODS GRAPHICS environment to get useful plots for the Airline Passenger example discussed previously in Example 11.2. Recall that when the ODS GRAPHICS environment is in effect, the printing options in the IDENTIFY, ESTIMATE, and FORECAST statements control the plotting behavior. In addition, the PLOT option must be on in the ESTIMATE statement in order to produce the residual correlation plots.

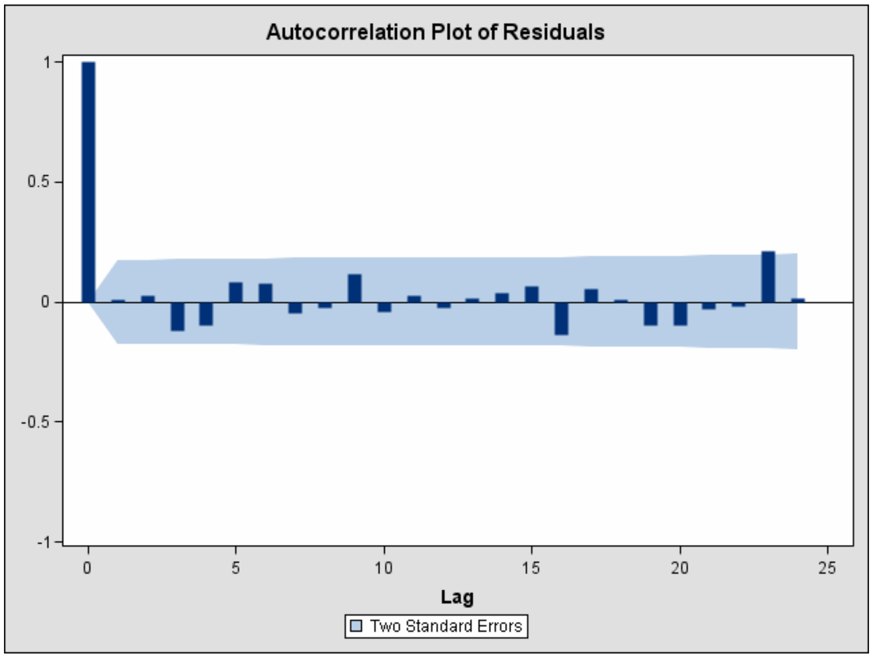
```
ods html;
ods graphics on;

proc arima data=seriesg;
   identify var=xlog(1 12);
   estimate q=(1)(12) noint plot;
   forecast printall id=date interval=month;
run;

ods graphics off;
ods html close;
```

Output 11.8.1 through Output 11.8.4 show a selection of the plots created.

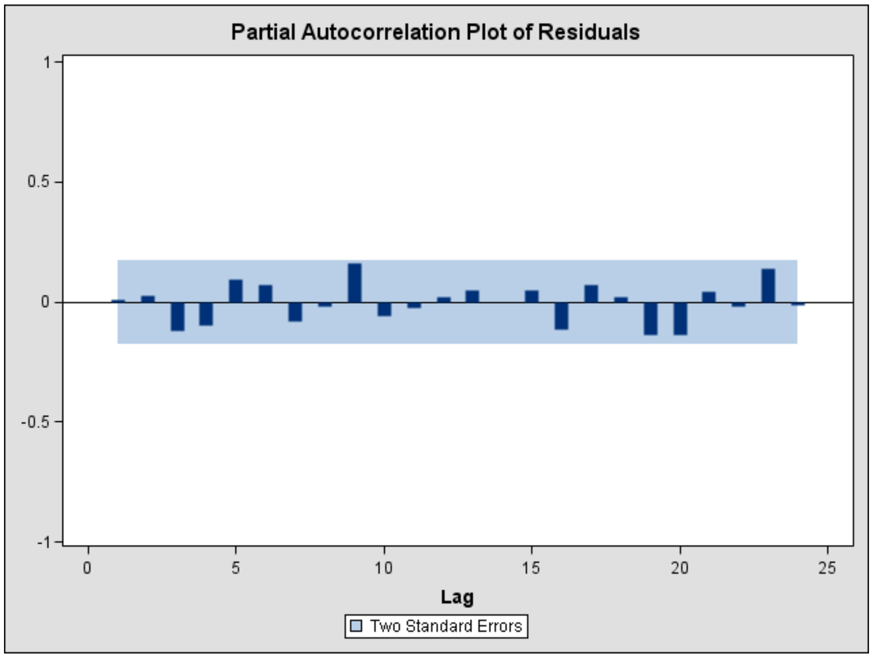
Output 11.8.1. ACF Plot for xlog(1 12) (Experimental)



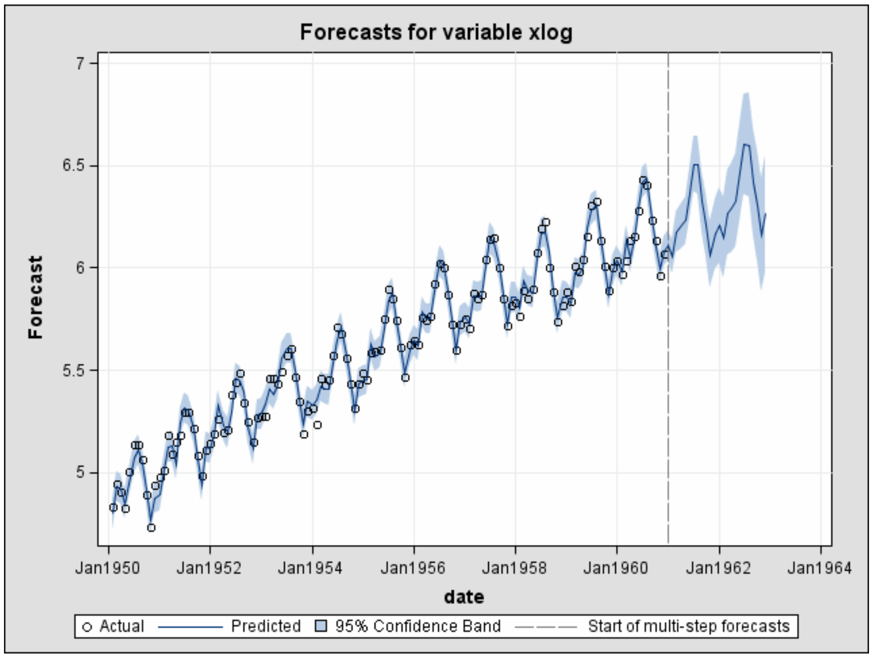
Output 11.8.2. Residual Autocorrelations for xlog (Experimental)



Output 11.8.3. Residual Partial Autocorrelations for xlog (Experimental)



Output 11.8.4. Forecast Plot (Experimental)



References

Akaike, H. (1974), “A New Look at the Statistical Model Identification,” *IEEE Transaction on Automatic Control*, AC-19, 716-723.

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, New York: John Wiley & Sons, Inc.

Andrews and Herzberg (1985), *A Collection Of Problems from Many Fields for the Student and Research Worker*, New York: Springer-Verlag.

Ansley, C. (1979), “An Algorithm for the Exact Likelihood of a Mixed Autoregressive-Moving Average Process,” *Biometrika*, 66, 59.

Ansley, C. and Newbold, P. (1980), “Finite Sample Properties of Estimators for Autoregressive Moving Average Models,” *Journal of Econometrics*, 13, 159.

Bhansali, R.J. (1980), “Autoregressive and Window Estimates of the Inverse Correlation Function,” *Biometrika*, 67, 551-566.

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.

Box, G.E.P., Jenkins, G.M., and Reinsel, G.C. (1994), *Time Series Analysis: Forecasting and Control*, Third Edition, Englewood Cliffs, NJ: Prentice Hall, 197-199.

Box, G.E.P. and Tiao, G.C. (1975), “Intervention Analysis with Applications to Economic and Environmental Problems,” *JASA*, 70, 70-79.

Brocklebank, J.C. and Dickey, D.A. (1986), *SAS System for Forecasting Time Series, 1986 Edition*, Cary, North Carolina: SAS Institute Inc.

Brockwell, P.J., and Davis, R.A. (1992), *Time Series: Theory and Methods*, Second Edition, New York: Springer-Verlag.

Chatfield, C. (1980), “Inverse Autocorrelations,” *Journal of the Royal Statistical Society*, A142, 363-377.

Choi, ByoungSeon (1992), *ARMA Model Identification*, New York: Springer-Verlag, 129-132.

Cleveland, W.S. (1972), “The Inverse Autocorrelations of a Time Series and Their Applications,” *Technometrics*, 14, 277.

Cobb, G.W. (1978), “The Problem of the Nile: Conditional Solution to a Change Point Problem,” *Biometrika*, 65, 243-251.

Davidson, J. (1981), “Problems with the Estimation of Moving Average Models,” *Journal of Econometrics*, 16, 295.

Davies, N., Triggs, C.M., and Newbold, P. (1977), “Significance Levels of the Box-Pierce Portmanteau Statistic in Finite Samples,” *Biometrika*, 64, 517-522.

de Jong, P. and Penzer, J. (1998), “Diagnosing Shocks in Time Series,” *Journal of the American Statistical Association*, Vol. 93, No. 442.

Dickey, D.A. (1976), "Estimation and Testing of Nonstationary Time Series," Unpublished Ph.D. Thesis, Iowa State University, Ames.

Dickey, D.A., and Fuller, W.A. (1979), "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, 74 (366), 427-431.

Dickey, D.A., Hasza, D.P., and Fuller, W.A. (1984), "Testing for Unit Roots in Seasonal Time Series," *Journal of the American Statistical Association*, 79 (386), 355-367.

Dunsmuir, William (1984), "Large Sample Properties of Estimation in Time Series Observed at Unequally Spaced Times," in *Time Series Analysis of Irregularly Observed Data*, Emanuel Parzen, ed., New York: Springer-Verlag.

Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C., and Chen, B.C. (1998), "New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program," *Journal of Business and Economic Statistics*, 16, 127-177.

Fuller, W.A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons, Inc.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton: Princeton University Press.

Hannan, E.J. and Rissanen, J. (1982), "Recursive Estimation of Mixed Autoregressive Moving Average Order," *Biometrika*, 69 (1), 81-94.

Harvey, A.C. (1981), *Time Series Models*, New York: John Wiley & Sons, Inc.

Jones, Richard H. (1980), "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations," *Technometrics*, 22, 389-396.

Kohn, R. and Ansley, C. (1985), "Efficient Estimation and Prediction in Time Series Regression Models," *Biometrika*, 72, 3, 694-697.

Ljung, G.M. and Box, G.E.P. (1978), "On a Measure of Lack of Fit in Time Series Models," *Biometrika*, 65, 297-303.

Montgomery, D.C. and Johnson, L.A. (1976), *Forecasting and Time Series Analysis*, New York: McGraw-Hill Book Co.

Morf, M., Sidhu, G.S., and Kailath, T. (1974), "Some New Algorithms for Recursive Estimation on Constant Linear Discrete Time Systems," *I.E.E.E. Transactions on Automatic Control*, AC-19, 315-323.

Nelson, C.R. (1973), *Applied Time Series for Managerial Forecasting*, San Francisco: Holden-Day.

Newbold, P. (1981), "Some Recent Developments in Time Series Analysis," *International Statistical Review*, 49, 53-66.

Newton, H. Joseph and Pagano, Marcello (1983), "The Finite Memory Prediction of Covariance Stationary Time Series," *SIAM Journal of Scientific and Statistical Computing*, 4, 330-339.

Pankratz, Alan (1983), *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*, New York: John Wiley & Sons, Inc.

Pankratz, Alan (1991), *Forecasting with Dynamic Regression Models*, New York: John Wiley & Sons, Inc.

Pearlman, J.G. (1980), "An Algorithm for the Exact Likelihood of a High-order Autoregressive-Moving Average Process," *Biometrika*, 67, 232-233.

Priestly, M.B. (1981), *Spectra Analysis and Time Series, Volume 1: Univariate Series*, New York: Academic Press, Inc.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461-464.

Stoffer, D. and Toloi, C. (1992), "A Note on the Ljung-Box-Pierce Portmanteau Statistic with Missing Data," *Statistics & Probability Letters* 13, 391-396.

Tsay, R.S. and Tiao, G.C. (1984), "Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation Function for Stationary and Nonstationary ARMA Models," *JASA*, 79 (385), 84-96.

Tsay, R.S. and Tiao, G.C. (1985), "Use of Canonical Analysis in Time Series Model Identification," *Biometrika*, 72 (2), 299-315.

Woodfield, T.J. (1987), "Time Series Intervention Analysis Using SAS Software," *Proceedings of the Twelfth Annual SAS Users Group International Conference*, 331-339. Cary, NC: SAS Institute Inc.

Chapter 12
The AUTOREG Procedure

Chapter Contents

Chapter 12
The AUTOREG Procedure

Overview

The AUTOREG procedure estimates and forecasts linear regression models for time series data when the errors are autocorrelated or heteroscedastic. The autoregressive error model is used to correct for autocorrelation, and the generalized autoregressive conditional heteroscedasticity (GARCH) model and its variants are used to model and correct for heteroscedasticity.

When time series data are used in regression analysis, often the error term is not independent through time. Instead, the errors are *serially correlated* or *autocorrelated*. If the error term is autocorrelated, the efficiency of ordinary least-squares (OLS) parameter estimates is adversely affected and standard error estimates are biased.

The autoregressive error model corrects for serial correlation. The AUTOREG procedure can fit autoregressive error models of any order and can fit subset autoregressive models. You can also specify stepwise autoregression to select the autoregressive error model automatically.

To diagnose autocorrelation, the AUTOREG procedure produces generalized Durbin-Watson (DW) statistics and their marginal probabilities. Exact *p*-values are reported for generalized DW tests to any specified order. For models with lagged dependent regressors, PROC AUTOREG performs the Durbin *t*-test and the Durbin *h*-test for first-order autocorrelation and reports their marginal significance levels.

Ordinary regression analysis assumes that the error variance is the same for all observations. When the error variance is not constant, the data are said to be *heteroscedastic*, and ordinary least-squares estimates are inefficient. Heteroscedasticity also affects the accuracy of forecast confidence limits. More efficient use of the data and more accurate prediction error estimates can be made by models that take the heteroscedasticity into account.

To test for heteroscedasticity, the AUTOREG procedure uses the portmanteau test statistics and the Engle Lagrange multiplier tests. Test statistics and significance *p*-values are reported for conditional heteroscedasticity at lags 1 through 12. The Bera-Jarque normality test statistic and its significance level are also reported to test for conditional nonnormality of residuals.

The family of GARCH models provides a means of estimating and correcting for the changing variability of the data. The GARCH process assumes that the errors, although uncorrelated, are not independent and models the conditional error variance as a function of the past realizations of the series.

The AUTOREG procedure supports the following variations of the GARCH models:

- generalized ARCH (GARCH)

- integrated GARCH (IGARCH)
- exponential GARCH (EGARCH)
- GARCH-in-mean (GARCH-M)

For GARCH-type models, the AUTOREG procedure produces the conditional prediction error variances as well as parameter and covariance estimates.

The AUTOREG procedure can also analyze models that combine autoregressive errors and GARCH-type heteroscedasticity. PROC AUTOREG can output predictions of the conditional mean and variance for models with autocorrelated disturbances and changing conditional error variances over time.

Four estimation methods are supported for the autoregressive error model:

- Yule-Walker
- iterated Yule-Walker
- unconditional least squares
- exact maximum likelihood

The maximum likelihood method is used for GARCH models and for mixed AR-GARCH models.

The AUTOREG procedure produces forecasts and forecast confidence limits when future values of the independent variables are included in the input data set. PROC AUTOREG is a useful tool for forecasting because it uses the time series part of the model as well as the systematic part in generating predicted values. The autoregressive error model takes into account recent departures from the trend in producing forecasts.

The AUTOREG procedure permits embedded missing values for the independent or dependent variables. The procedure should be used only for ordered and equally spaced time series data.

Experimental graphics are now available with the AUTOREG procedure. For more information, see the "ODS Graphics" section on page 560.

Getting Started

Regression with Autocorrelated Errors

Ordinary regression analysis is based on several statistical assumptions. One key assumption is that the errors are independent of each other. However, with time series data, the ordinary regression residuals usually are correlated over time. It is not desirable to use ordinary regression analysis for time series data since the assumptions on which the classical linear regression model is based will usually be violated.

Violation of the independent errors assumption has three important consequences for ordinary regression. First, statistical tests of the significance of the parameters and the confidence limits for the predicted values are not correct. Second, the estimates of the regression coefficients are not as efficient as they would be if the autocorrelation were taken into account. Third, since the ordinary regression residuals are not independent, they contain information that can be used to improve the prediction of future values.

The AUTOREG procedure solves this problem by augmenting the regression model with an autoregressive model for the random error, thereby accounting for the autocorrelation of the errors. Instead of the usual regression model, the following autoregressive error model is used:

$$y_t = \mathbf{x}_t'\beta + \nu_t$$
$$\nu_t = -\varphi_1\nu_{t-1} - \varphi_2\nu_{t-2} - \ldots - \varphi_m\nu_{t-m} + \epsilon_t$$
$$\epsilon_t \sim \mathrm{IN}(0, \sigma^2)$$

The notation $\epsilon_t \sim \mathrm{IN}(0, \sigma^2)$ indicates that each ϵ_t is normally and independently distributed with mean 0 and variance σ^2.

By simultaneously estimating the regression coefficients β and the autoregressive error model parameters φ_i, the AUTOREG procedure corrects the regression estimates for autocorrelation. Thus, this kind of regression analysis is often called *autoregressive error correction* or *serial correlation correction*.

Example of Autocorrelated Data

A simulated time series is used to introduce the AUTOREG procedure. The following statements generate a simulated time series Y with second-order autocorrelation:

```
data a;
   ul = 0; ull = 0;
   do time = -10 to 36;
      u = + 1.3 * ul - .5 * ull + 2*rannor(12346);
      y = 10 + .5 * time + u;
      if time > 0 then output;
      ull = ul; ul = u;
      end;
run;
```

The series Y is a time trend plus a second-order autoregressive error. The model simulated is

$$y_t = 10 + .5t + \nu_t$$
$$\nu_t = 1.3\nu_{t-1} - .5\nu_{t-2} + \epsilon_t$$
$$\epsilon_t \sim \mathrm{IN}(0, 4)$$

The following statements plot the simulated time series Y. A linear regression trend line is shown for reference. (The regression line is produced by plotting the series a second time using the regression interpolation feature of the SYMBOL statement. Refer to *SAS/GRAPH Software: Reference, Version 6, First Edition, Volume 1* for further explanation.)

```
title \quotes{Autocorrelated Time Series};
proc gplot data=a;
```

```
      symbol1 v=dot i=join;
      symbol2 v=none i=r;
      plot y * time = 1 y * time = 2 / overlay;
   run;
```

The plot of series Y and the regression line are shown in Figure 12.1.

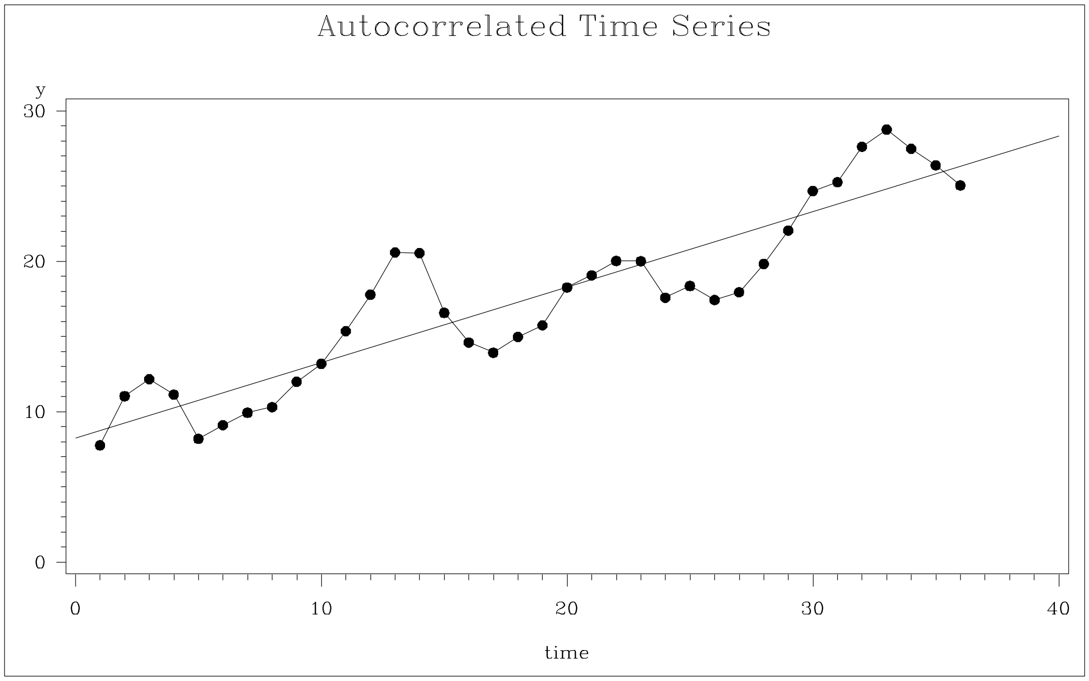


Figure 12.1. Autocorrelated Time Series

Note that when the series is above (or below) the OLS regression trend line, it tends to remain above (below) the trend for several periods. This pattern is an example of *positive autocorrelation*.

Time series regression usually involves independent variables other than a time-trend. However, the simple time-trend model is convenient for illustrating regression with autocorrelated errors, and the series Y shown in Figure 12.1 is used in the following introductory examples.

Ordinary Least-Squares Regression

To use the AUTOREG procedure, specify the input data set in the PROC AUTOREG statement and specify the regression model in a MODEL statement. Specify the model by first naming the dependent variable and then listing the regressors after an equal sign, as is done in other SAS regression procedures. The following statements regress Y on TIME using ordinary least squares:

```
   proc autoreg data=a;
      model y = time;
   run;
```

The AUTOREG procedure output is shown in Figure 12.2.

```
                          The AUTOREG Procedure

                        Dependent Variable    y


                    Ordinary Least Squares Estimates

    SSE                   214.953429    DFE                      34
    MSE                      6.32216    Root MSE            2.51439
    SBC                   173.659101    AIC              170.492063
    Regress R-Square          0.8200    Total R-Square       0.8200
    Durbin-Watson             0.4752


                                        Standard                 Approx
 Variable        DF      Estimate          Error    t Value    Pr > |t|

 Intercept        1        8.2308         0.8559       9.62      <.0001
 time             1        0.5021         0.0403      12.45      <.0001
```

Figure 12.2. AUTOREG Results for OLS estimation

The output first shows statistics for the model residuals. The model root mean square error (Root MSE) is 2.51, and the model R^2 is .82. Notice that two R^2 statistics are shown, one for the regression model (Reg Rsq) and one for the full model (Total Rsq) that includes the autoregressive error process, if any. In this case, an autoregressive error model is not used, so the two R^2 statistics are the same.

Other statistics shown are the sum of square errors (SSE), mean square error (MSE), error degrees of freedom (DFE, the number of observations minus the number of parameters), the information criteria SBC and AIC, and the Durbin-Watson statistic. (Durbin-Watson statistics and SBC and AIC are discussed in the "Details" section later in this chapter.)

The output then shows a table of regression coefficients, with standard errors and *t*-tests. The estimated model is

$$y_t = 8.23 + .502t + \epsilon_t$$
$$Est.\ Var(\epsilon_t) = 6.32$$

The OLS parameter estimates are reasonably close to the true values, but the estimated error variance, 6.32, is much larger than the true value, 4.

Autoregressive Error Model

The following statements regress Y on TIME with the errors assumed to follow a second-order autoregressive process. The order of the autoregressive model is specified by the NLAG=2 option. The Yule-Walker estimation method is used by default. The example uses the METHOD=ML option to specify the exact maximum likelihood method instead.

```
proc autoreg data=a;
   model y = time / nlag=2 method=ml;
   run;
```

The first part of the results are shown in Figure 12.3. The initial OLS results are produced first, followed by estimates of the autocorrelations computed from the OLS residuals. The autocorrelations are also displayed graphically.

```
                              The AUTOREG Procedure

                            Dependent Variable    y


                        Ordinary Least Squares Estimates

         SSE                   214.953429    DFE                        34
         MSE                      6.32216    Root MSE              2.51439
         SBC                   173.659101    AIC                170.492063
         Regress R-Square          0.8200    Total R-Square         0.8200
         Durbin-Watson             0.4752


                                             Standard                 Approx
      Variable        DF       Estimate         Error    t Value    Pr > |t|

      Intercept        1         8.2308        0.8559       9.62      <.0001
      time             1         0.5021        0.0403      12.45      <.0001


                          Estimates of Autocorrelations

   Lag    Covariance    Correlation    -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

     0        5.9709       1.000000    |                    |********************|
     1        4.5169       0.756485    |                    |***************     |
     2        2.0241       0.338995    |                    |*******             |


                           Preliminary MSE      1.7943
```

Figure 12.3. Preliminary Estimate for AR(2) Error Model

The maximum likelihood estimates are shown in Figure 12.4. Figure 12.4 also shows the preliminary Yule-Walker estimates used as starting values for the iterative computation of the maximum likelihood estimates.

```
                         The AUTOREG Procedure

                 Estimates of Autoregressive Parameters

                                             Standard
             Lag      Coefficient               Error     t Value

               1        -1.169057            0.148172       -7.89
               2         0.545379            0.148172        3.68


    Algorithm converged.


                      Maximum Likelihood Estimates

     SSE                   54.7493022    DFE                     32
     MSE                      1.71092    Root MSE           1.30802
     SBC                   133.476508    AIC             127.142432
     Regress R-Square          0.7280    Total R-Square      0.9542
     Durbin-Watson             2.2761


                                          Standard                 Approx
   Variable        DF      Estimate          Error     t Value    Pr > |t|

   Intercept        1        7.8833         1.1693        6.74      <.0001
   time             1        0.5096         0.0551        9.25      <.0001
   AR1              1       -1.2464         0.1385       -9.00      <.0001
   AR2              1        0.6283         0.1366        4.60      <.0001


                 Autoregressive parameters assumed given.

                                          Standard                 Approx
   Variable        DF      Estimate          Error     t Value    Pr > |t|

   Intercept        1        7.8833         1.1678        6.75      <.0001
   time             1        0.5096         0.0551        9.26      <.0001
```

Figure 12.4. Maximum Likelihood Estimates of AR(2) Error Model

The diagnostic statistics and parameter estimates tables in Figure 12.4 have the same form as in the OLS output, but the values shown are for the autoregressive error model. The MSE for the autoregressive model is 1.71, which is much smaller than the true value of 4. In small samples, the autoregressive error model tends to underestimate σ^2, while the OLS MSE overestimates σ^2.

Notice that the total R^2 statistic computed from the autoregressive model residuals is .954, reflecting the improved fit from the use of past residuals to help predict the next Y value. The Reg Rsq value .728 is the R^2 statistic for a regression of transformed variables adjusted for the estimated autocorrelation. (This is not the R^2 for the estimated trend line. For details, see "R^2 Statistics and Other Measures of Fit" later in this chapter.)

The parameter estimates table shows the ML estimates of the regression coefficients and includes two additional rows for the estimates of the autoregressive parameters, labeled A(1) and A(2).

The estimated model is

$$y_t = 7.88 + .5096t + \nu_t$$

$$\nu_t = 1.25\nu_{t-1} - .628\nu_{t-2} + \epsilon_t$$

$$Est.\ Var(\epsilon_t) = 1.71$$

Note that the signs of the autoregressive parameters shown in this equation for ν_t are the reverse of the estimates shown in the AUTOREG procedure output. Figure 12.4 also shows the estimates of the regression coefficients with the standard errors recomputed on the assumption that the autoregressive parameter estimates equal the true values.

Predicted Values and Residuals

The AUTOREG procedure can produce two kinds of predicted values and corresponding residuals and confidence limits. The first kind of predicted value is obtained from only the structural part of the model, $\mathbf{x}_t'\mathbf{b}$. This is an estimate of the unconditional mean of the response variable at time *t*. For the time trend model, these predicted values trace the estimated trend. The second kind of predicted values include both the structural part of the model and the predicted values of the autoregressive error process. The full model (conditional) predictions are used to forecast future values.

Use the OUTPUT statement to store predicted values and residuals in a SAS data set and to output other values such as confidence limits and variance estimates. The P= option specifies an output variable to contain the full model predicted values. The PM= option names an output variable for the predicted mean. The R= and RM= options specify output variables for the corresponding residuals, computed as the actual value minus the predicted value.

The following statements store both kinds of predicted values in the output data set. (The printed output is the same as previously shown in Figure 12.3 and Figure 12.4.)

```
proc autoreg data=a;
   model y = time / nlag=2 method=ml;
   output out=p p=yhat pm=trendhat;
run;
```

The following statements plot the predicted values from the regression trend line and from the full model together with the actual values.

```
title \quotes{Predictions for Autocorrelation Model};
proc gplot data=p;
   symbol1 v=star i=none;
   symbol2 v=circle i=join;
   symbol3 v=none i=join;
   plot y * time = 1 yhat * time = 2
        trendhat * time = 3 / overlay ;
run;
```

The plot of predicted values is shown in Figure 12.5.

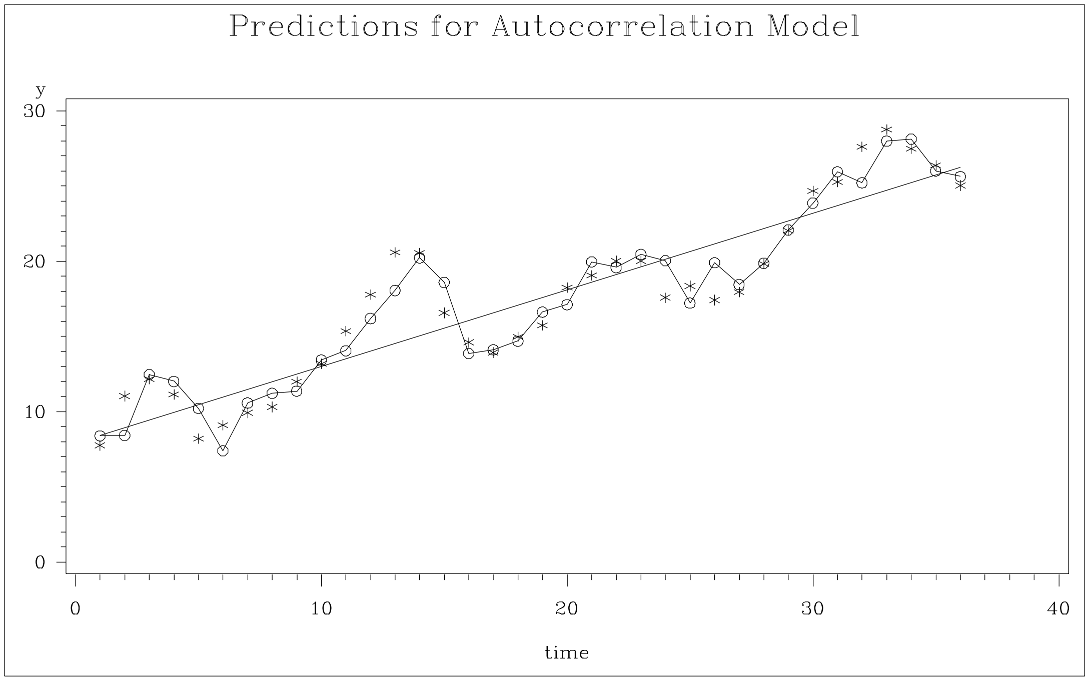


Figure 12.5. PROC AUTOREG Predictions

In Figure 12.5 the straight line is the autocorrelation corrected regression line, traced out by the structural predicted values TRENDHAT. The jagged line traces the full model prediction values. The actual values are marked by asterisks. This plot graphically illustrates the improvement in fit provided by the autoregressive error process for highly autocorrelated data.

Forecasting Autoregressive Error Models

To produce forecasts for future periods, include observations for the forecast periods in the input data set. The forecast observations must provide values for the independent variables and have missing values for the response variable.

For the time trend model, the only regressor is time. The following statements add observations for time periods 37 through 46 to the data set A to produce an augmented data set B:

```
data b;
   y = .;
   do time = 37 to 46; output; end;
run;
```

```
   data b; merge a b; by time; run;
```

To produce the forecast, use the augmented data set as input to PROC AUTOREG, and specify the appropriate options in the OUTPUT statement. The following statements produce forecasts for the time trend with autoregressive error model. The output data set includes all the variables in the input data set, the forecast values (YHAT), the predicted trend (YTREND), and the upper (UCL) and lower (LCL) 95% confidence limits.

```
   proc autoreg data=b;
      model y = time / nlag=2 method=ml;
      output out=p p=yhat pm=ytrend
                   lcl=lcl ucl=ucl;
   run;
```

The following statements plot the predicted values and confidence limits, and they also plot the trend line for reference. The actual observations are shown for periods 16 through 36, and a reference line is drawn at the start of the out-of-sample forecasts.

```
   title \quotes{Forecasting Autocorrelated Time Series};
   proc gplot data=p;
      plot y*time=1 yhat*time=2 ytrend*time=3
           lcl*time=3 ucl*time=3 /
           overlay href=36.5;
      where time >= 16;
      symbol1 v=star i=none;
      symbol2 v=circle i=join;
      symbol3 v=none i=join;
   run;
```

The plot is shown in Figure 12.6. Notice that the forecasts take into account the recent departures from the trend but converge back to the trend line for longer forecast horizons.

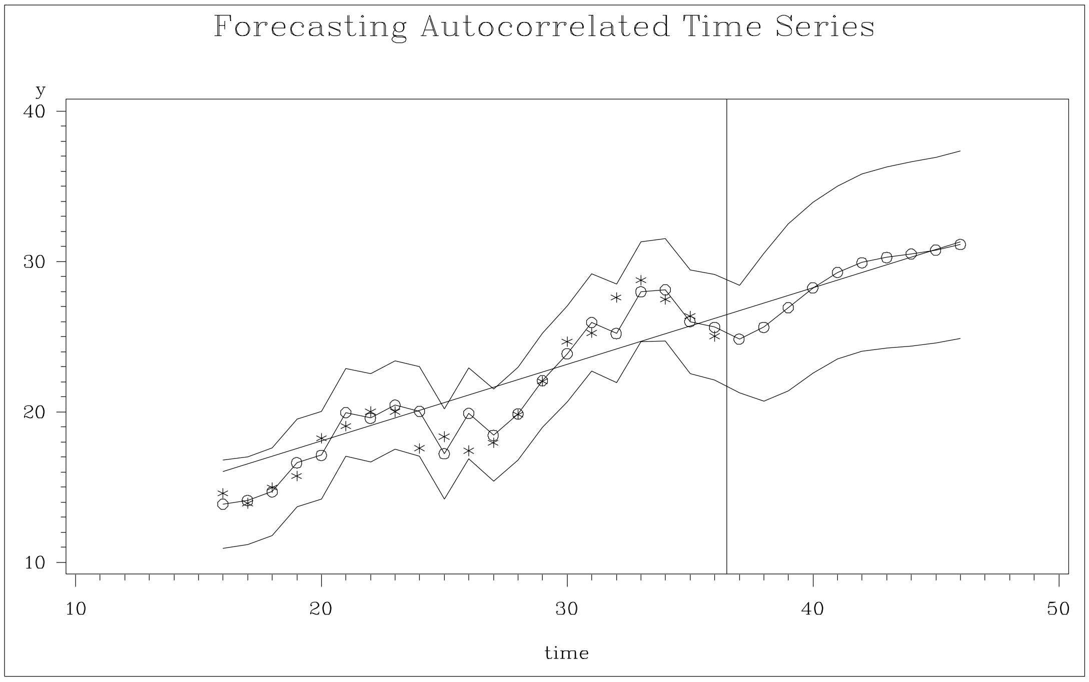


Figure 12.6. PROC AUTOREG Forecasts

Testing for Autocorrelation

In the preceding section, it is assumed that the order of the autoregressive process is known. In practice, you need to test for the presence of autocorrelation.

The Durbin-Watson test is a widely used method of testing for autocorrelation. The first-order Durbin-Watson statistic is printed by default. This statistic can be used to test for first-order autocorrelation. Use the DWPROB option to print the significance level (*p*-values) for the Durbin-Watson tests. (Since the Durbin-Watson *p*-values are computationally expensive, they are not reported by default.)

You can use the DW= option to request higher-order Durbin-Watson statistics. Since the ordinary Durbin-Watson statistic only tests for first-order autocorrelation, the Durbin-Watson statistics for higher-order autocorrelation are called *generalized Durbin-Watson statistics*.

The following statements perform the Durbin-Watson test for autocorrelation in the OLS residuals for orders 1 through 4. The DWPROB option prints the marginal significance levels (*p*-values) for the Durbin-Watson statistics.

```
proc autoreg data=a;
   model y = time / dw=4 dwprob;
run;
```

The AUTOREG procedure output is shown in Figure 12.7. In this case, the first-order Durbin-Watson test is highly significant, with $p < .0001$ for the hypothesis of no first-order autocorrelation. Thus, autocorrelation correction is needed.

```
                              The AUTOREG Procedure

                             Dependent Variable    y


                         Ordinary Least Squares Estimates

          SSE                  214.953429    DFE                     34
          MSE                     6.32216    Root MSE           2.51439
          SBC                  173.659101    AIC             170.492063
          Regress R-Square         0.8200    Total R-Square      0.8200


                            Durbin-Watson Statistics

                       Order           DW    Pr < DW    Pr > DW

                          1        0.4752     <.0001     1.0000
                          2        1.2935     0.0137     0.9863
                          3        2.0694     0.6545     0.3455
                          4        2.5544     0.9818     0.0182
NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is
      the p-value for testing negative autocorrelation.


                                            Standard                 Approx
       Variable        DF     Estimate         Error    t Value    Pr > |t|

       Intercept        1       8.2308        0.8559       9.62      <.0001
       time             1       0.5021        0.0403      12.45      <.0001
```

Figure 12.7. Durbin-Watson Test Results for OLS Residuals

Using the Durbin-Watson test, you can decide if autocorrelation correction is needed. However, generalized Durbin-Watson tests should not be used to decide on the autoregressive order. The higher-order tests assume the absence of lower-order autocorrelation. If the ordinary Durbin-Watson test indicates no first-order autocorrelation, you can use the second-order test to check for second-order autocorrelation. Once autocorrelation is detected, further tests at higher orders are not appropriate. In Figure 12.7, since the first-order Durbin-Watson test is significant, the order 2, 3, and 4 tests can be ignored.

When using Durbin-Watson tests to check for autocorrelation, you should specify an order at least as large as the order of any potential seasonality, since seasonality produces autocorrelation at the seasonal lag. For example, for quarterly data use DW=4, and for monthly data use DW=12.

Lagged Dependent Variables

The Durbin-Watson tests are not valid when the lagged dependent variable is used in the regression model. In this case, the Durbin *h*-test or Durbin *t*-test can be used to test for first-order autocorrelation.

For the Durbin *h*-test, specify the name of the lagged dependent variable in the LAGDEP= option. For the Durbin *t*-test, specify the LAGDEP option without giving the name of the lagged dependent variable.

For example, the following statements add the variable YLAG to the data set A and regress Y on YLAG instead of TIME.

```
data b;
   set a;
   ylag = lag1( y );
run;

proc autoreg data=b;
   model y = ylag / lagdep=ylag;
run;
```

The results are shown in Figure 12.8. The Durbin *h* statistic 2.78 is significant with a *p*-value of .0027, indicating autocorrelation.

```
                           The AUTOREG Procedure

                          Dependent Variable    y


                      Ordinary Least Squares Estimates

     SSE                    97.711226    DFE                       33
     MSE                      2.96095    Root MSE             1.72074
     SBC                   142.369787    AIC               139.259091
     Regress R-Square          0.9109    Total R-Square        0.9109
     Durbin h                  2.7814    Pr > h                0.0027


                                           Standard                Approx
  Variable        DF      Estimate          Error     t Value    Pr > |t|

  Intercept        1        1.5742         0.9300        1.69      0.0999
  ylag             1        0.9376         0.0510       18.37      <.0001
```

Figure 12.8. Durbin *h* -Test With a Lagged Dependent Variable

Stepwise Autoregression

Once you determine that autocorrelation correction is needed, you must select the order of the autoregressive error model to use. One way to select the order of the autoregressive error model is *stepwise autoregression*. The stepwise autoregression method initially fits a high-order model with many autoregressive lags and then sequentially removes autoregressive parameters until all remaining autoregressive parameters have significant *t*-tests.

To use stepwise autoregression, specify the BACKSTEP option, and specify a large order with the NLAG= option. The following statements show the stepwise feature, using an initial order of 5:

```
proc autoreg data=a;
   model y = time / method=ml nlag=5 backstep;
run;
```

The results are shown in Figure 12.9.

```
                              The AUTOREG Procedure

                             Dependent Variable    y


                         Ordinary Least Squares Estimates

         SSE                   214.953429    DFE                      34
         MSE                      6.32216    Root MSE            2.51439
         SBC                   173.659101    AIC              170.492063
         Regress R-Square          0.8200    Total R-Square       0.8200
         Durbin-Watson             0.4752


                                           Standard                Approx
      Variable         DF     Estimate        Error    t Value    Pr > |t|

      Intercept         1       8.2308       0.8559       9.62      <.0001
      time              1       0.5021       0.0403      12.45      <.0001


                          Estimates of Autocorrelations

   Lag   Covariance    Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

     0       5.9709       1.000000   |                    |********************|
     1       4.5169       0.756485   |                    |***************     |
     2       2.0241       0.338995   |                    |*******             |
     3      -0.4402      -0.073725   |                   *|                    |
     4      -2.1175      -0.354632   |             *******|                    |
     5      -2.8534      -0.477887   |          **********|                    |


                              Backward Elimination of
                               Autoregressive Terms

                     Lag      Estimate     t Value    Pr > |t|

                       4     -0.052908       -0.20      0.8442
                       3      0.115986        0.57      0.5698
                       5      0.131734        1.21      0.2340
```

Figure 12.9. Stepwise Autoregression

The estimates of the autocorrelations are shown for 5 lags. The backward elimination of autoregressive terms report shows that the autoregressive parameters at lags 3, 4, and 5 were insignificant and eliminated, resulting in the second-order model shown previously in Figure 12.4. By default, retained autoregressive parameters must be significant at the .05 level, but you can control this with the SLSTAY= option. The remainder of the output from this example is the same as that in Figure 12.3 and Figure 12.4, and it is not repeated here.

The stepwise autoregressive process is performed using the Yule-Walker method. The maximum likelihood estimates are produced after the order of the model is determined from the significance tests of the preliminary Yule-Walker estimates.

When using stepwise autoregression, it is a good idea to specify an NLAG= option value larger than the order of any potential seasonality, since seasonality produces autocorrelation at the seasonal lag. For example, for monthly data use NLAG=13, and for quarterly data use NLAG=5.

Subset and Factored Models

In the previous example, the BACKSTEP option dropped lags 3, 4, and 5, leaving an order 2 model. However, in other cases a parameter at a longer lag may be kept while some smaller lags are dropped. For example, the stepwise autoregression method might drop lags 2, 3, and 5 but keep lags 1 and 4. This is called a *subset model*, since the number of estimated autoregressive parameters is smaller than the order of the model.

Subset models are common for seasonal data and often correspond to *factored* autoregressive models. A factored model is the product of simpler autoregressive models. For example, the best model for seasonal monthly data may be the combination of a first-order model for recent effects with a twelfth-order subset model for the seasonality, with a single parameter at lag 12. This results in an order 13 subset model with nonzero parameters at lags 1, 12, and 13. See Chapter 11, “The ARIMA Procedure,” for further discussion of subset and factored autoregressive models.

You can specify subset models with the NLAG= option. List the lags to include in the autoregressive model within parentheses. The following statements show an example of specifying the subset model resulting from the combination of a first-order process for recent effects with a fourth-order seasonal process:

```
proc autoreg data=a;
   model y = time / nlag=(1 4 5);
run;
```

The MODEL statement specifies the following fifth-order autoregressive error model:

$$y_t = a + bt + \nu_t$$

$$\nu_t = -\varphi_1\nu_{t-1} - \varphi_4\nu_{t-4} - \varphi_5\nu_{t-5} + \epsilon_t$$

Testing for Heteroscedasticity

One of the key assumptions of the ordinary regression model is that the errors have the same variance throughout the sample. This is also called the *homoscedasticity* model. If the error variance is not constant, the data are said to be *heteroscedastic*.

Since ordinary least-squares regression assumes constant error variance, heteroscedasticity causes the OLS estimates to be inefficient. Models that take into account the changing variance can make more efficient use of the data. Also, heteroscedasticity can make the OLS forecast error variance inaccurate since the predicted forecast variance is based on the average variance instead of the variability at the end of the series.

To illustrate heteroscedastic time series, the following statements re-create the simulated series Y. The variable Y has an error variance that changes from 1 to 4 in the middle part of the series. The length of the series is also extended 120 observations.

```
data a;
   ul = 0; ull = 0;
   do time = -10 to 120;
      s = 1 + (time >= 60 & time < 90);
      u = + 1.3 * ul - .5 * ull + s*rannor(12346);
      y = 10 + .5 * time + u;
      if time > 0 then output;
      ull = ul; ul = u;
      end;
run;

title \quotes{Heteroscedastic Autocorrelated Time Series};
proc gplot data=a;
   symbol1 v=dot i=join;
   symbol2 v=none i=r;
   plot y * time = 1 y * time = 2 / overlay;
run;
```

The simulated series is plotted in Figure 12.10.

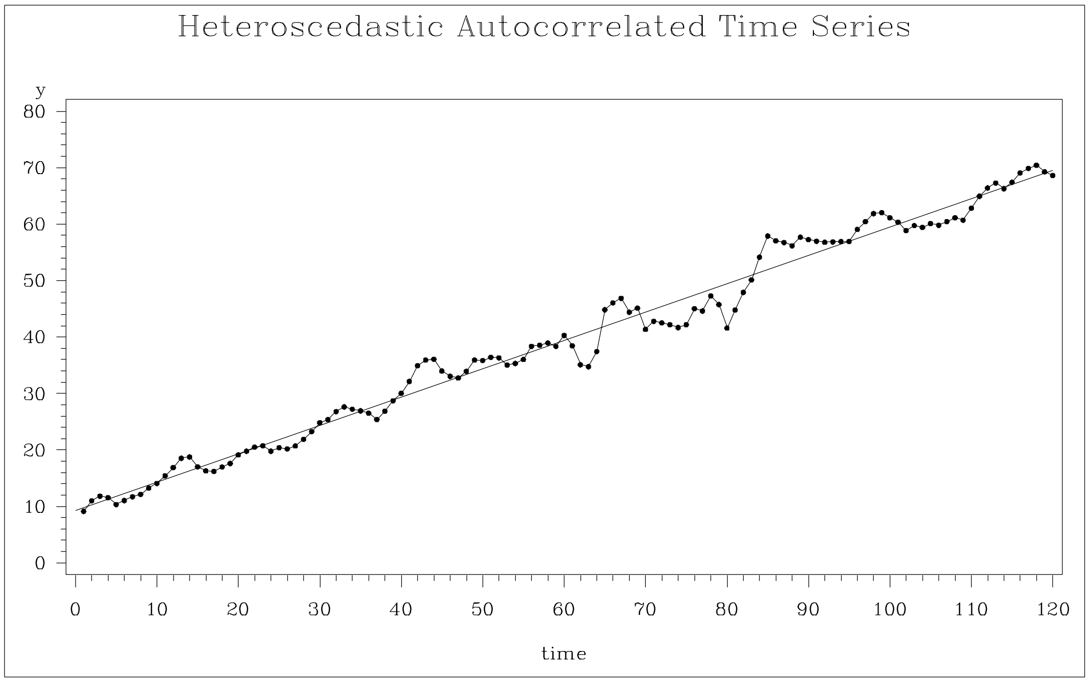


Figure 12.10. Heteroscedastic and Autocorrelated Series

To test for heteroscedasticity with PROC AUTOREG, specify the ARCHTEST option. The following statements regress Y on TIME and use the ARCHTEST option to test for heteroscedastic OLS residuals. The DWPROB option is also used to test for autocorrelation.

```
proc autoreg data=a;
   model y = time / nlag=2 archtest dwprob;
   output out=r r=yresid;
run;
```

The PROC AUTOREG output is shown in Figure 12.11. The Q statistics test for changes in variance across time using lag windows ranging from 1 through 12. (See "Heteroscedasticity and Normality Tests" for details.) The *p*-values for the test statistics are given in parentheses. These tests strongly indicate heteroscedasticity, with $p < 0.0001$ for all lag windows.

The Lagrange multiplier (LM) tests also indicate heteroscedasticity. These tests can also help determine the order of the ARCH model appropriate for modeling the heteroscedasticity, assuming that the changing variance follows an autoregressive conditional heteroscedasticity model.

```
                              The AUTOREG Procedure

                             Dependent Variable    y


                         Ordinary Least Squares Estimates

          SSE                  690.266009    DFE                      118
          MSE                     5.84971    Root MSE             2.41862
          SBC                  560.070468    AIC               554.495484
          Regress R-Square         0.9814    Total R-Square        0.9814
          Durbin-Watson            0.4060    Pr < DW               <.0001
          Pr > DW                  1.0000


                       Q and LM Tests for ARCH Disturbances

              Order            Q     Pr > Q           LM     Pr > LM

                  1      37.5445     <.0001      37.0072      <.0001
                  2      40.4245     <.0001      40.9189      <.0001
                  3      41.0753     <.0001      42.5032      <.0001
                  4      43.6893     <.0001      43.3822      <.0001
                  5      55.3846     <.0001      48.2511      <.0001
                  6      60.6617     <.0001      49.7799      <.0001
                  7      62.9655     <.0001      52.0126      <.0001
                  8      63.7202     <.0001      52.7083      <.0001
                  9      64.2329     <.0001      53.2393      <.0001
                 10      66.2778     <.0001      53.2407      <.0001
                 11      68.1923     <.0001      53.5924      <.0001
                 12      69.3725     <.0001      53.7559      <.0001


                                            Standard                  Approx
       Variable        DF      Estimate        Error     t Value     Pr > |t|

       Intercept        1        9.2217       0.4444       20.75       <.0001
       time             1        0.5024     0.006374       78.83       <.0001
```

Figure 12.11. Heteroscedasticity Tests

Heteroscedasticity and GARCH Models

There are several approaches to dealing with heteroscedasticity. If the error variance at different times is known, weighted regression is a good method. If, as is usually the case, the error variance is unknown and must be estimated from the data, you can model the changing error variance.

The *generalized autoregressive conditional heteroscedasticity* (GARCH) model is one approach to modeling time series with heteroscedastic errors. The GARCH regression model with autoregressive errors is

$$y_t = \mathbf{x}_t'\beta + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1\nu_{t-1} - \ldots - \varphi_m\nu_{t-m}$$

$$\epsilon_t = \sqrt{h_t}e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

$$e_t \sim \mathrm{IN}(0,1)$$

This model combines the *m*th-order autoregressive error model with the GARCH(*p*,*q*) variance model. It is denoted as the AR(*m*)-GARCH(*p*,*q*) regression model.

The Lagrange multiplier (LM) tests shown in Figure 12.11 can help determine the order of the ARCH model appropriate for the data. The tests are significant (p<.0001) through order 12, which indicates that a very high-order ARCH model is needed to model the heteroscedasticity.

The basic ARCH(*q*) model (*p*=0) is a *short memory* process in that only the most recent *q* squared residuals are used to estimate the changing variance. The GARCH model (*p*>0) allows *long memory* processes, which use all the past squared residuals to estimate the current variance. The LM tests in Figure 12.11 suggest the use of the GARCH model (*p*>0) instead of the ARCH model.

The GARCH(*p*,*q*) model is specified with the GARCH=(P=*p*,Q=*q*) option in the MODEL statement. The basic ARCH(*q*) model is the same as the GARCH(0,*q*) model and is specified with the GARCH=(Q=*q*) option.

The following statements fit an AR(2)-GARCH(1,1) model for the Y series regressed on TIME. The GARCH=(P=1,Q=1) option specifies the GARCH(1,1) conditional variance model. The NLAG=2 option specifies the AR(2) error process. Only the maximum likelihood method is supported for GARCH models; therefore, the METHOD= option is not needed. The CEV= option in the OUTPUT statement stores the estimated conditional error variance at each time period in the variable VHAT in an output data set named OUT.

```
proc autoreg data=a;
   model y = time / nlag=2 garch=(q=1,p=1) maxit=50;
   output out=out cev=vhat;
run;
```

The results for the GARCH model are shown in Figure 12.12. (The preliminary estimates are not shown.)

```
                             The AUTOREG Procedure

                                GARCH Estimates

          SSE                218.860964    Observations             120
          MSE                   1.82384    Uncond Var        1.62996534
          Log Likelihood     -187.44013    Total R-Square        0.9941
          SBC                408.392693    AIC                388.88025
          Normality Test         0.0839    Pr > ChiSq            0.9589


                                            Standard                 Approx
      Variable        DF     Estimate        Error    t Value    Pr > |t|

      Intercept        1       8.9301       0.7235      12.34      <.0001
      time             1       0.5075       0.0107      47.30      <.0001
      AR1              1      -1.2301       0.1078     -11.41      <.0001
      AR2              1       0.5023       0.1057       4.75      <.0001
      ARCH0            1       0.0850       0.0757       1.12      0.2614
      ARCH1            1       0.2103       0.0847       2.48      0.0130
      GARCH1           1       0.7376       0.0960       7.68      <.0001
```

Figure 12.12. AR(2)-GARCH(1, 1) Model

The normality test is not significant ($p = 0.957$), which is consistent with the hypothesis that the residuals from the GARCH model, $\epsilon_t/\sqrt{h_t}$, are normally distributed. The parameter estimates table includes rows for the GARCH parameters. ARCH0 represents the estimate for the parameter ω, ARCH1 represents α_1, and GARCH1 represents γ_1.

The following statements transform the estimated conditional error variance series VHAT to the estimated standard deviation series SHAT. Then, they plot SHAT together with the true standard deviation S used to generate the simulated data.

```
data out;
   set out;
   shat = sqrt( vhat );
run;

title \quotes{Predicted and Actual Standard Deviations};
proc gplot data=out;
   plot s*time=1 shat*time=2 / overlay;
   symbol1 v=dot  i=none;
   symbol2 v=none i = join;
run;
```

The plot is shown in Figure 12.13.

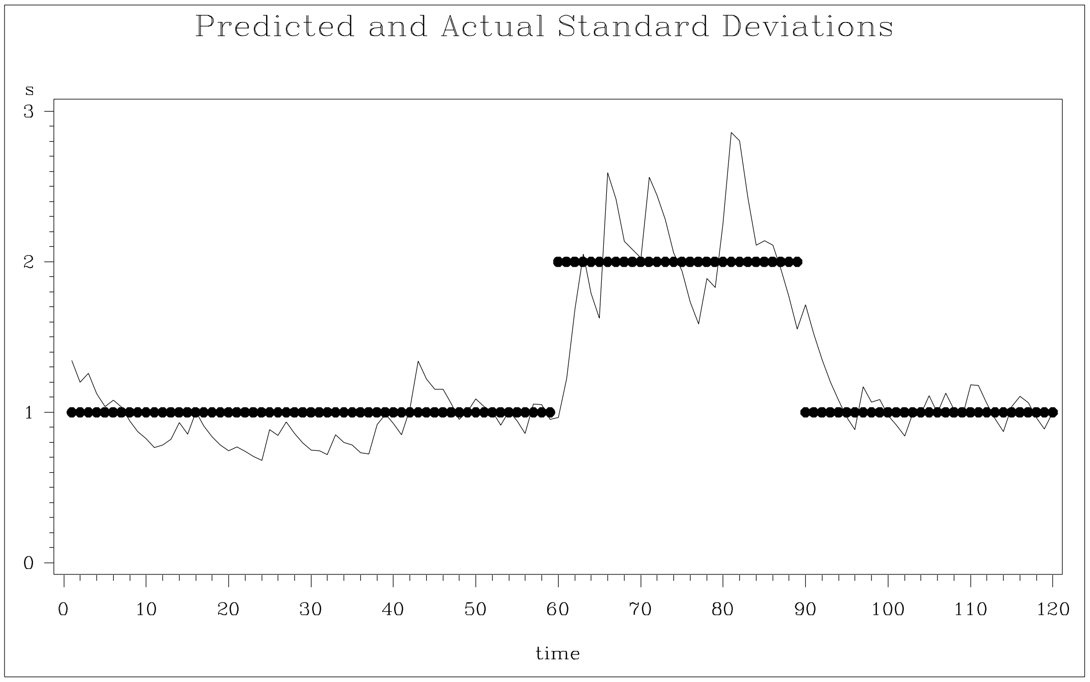


Figure 12.13. Estimated and Actual Error Standard Deviation Series

Note that in this example the form of heteroscedasticity used in generating the simulated series Y does not fit the GARCH model. The GARCH model assumes *conditional* heteroscedasticity, with homoscedastic unconditional error variance. That is, the GARCH model assumes that the changes in variance are a function of the realizations of preceding errors and that these changes represent temporary and random departures from a constant unconditional variance. The data generating process used to simulate series Y, contrary to the GARCH model, has exogenous unconditional heteroscedasticity that is independent of past errors.

Nonetheless, as shown in Figure 12.13, the GARCH model does a reasonably good job of approximating the error variance in this example, and some improvement in the efficiency of the estimator of the regression parameters can be expected.

The GARCH model may perform better in cases where theory suggests that the data generating process produces true autoregressive conditional heteroscedasticity. This is the case in some economic theories of asset returns, and GARCH-type models are often used for analysis of financial markets data.

Using the HETERO Statement with GARCH Models

The HETERO statement can be combined with the GARCH= option on the MODEL statement to include input variables in the GARCH conditional variance model. For example, the GARCH(1,1) variance model with two dummy input variables D1 and D2 is

$$\epsilon_t = \sqrt{h_t} e_t$$

$$h_t = \omega + \alpha_1 \epsilon_{t-1}^2 + \gamma_1 h_{t-1} + \eta_1 D1_t + \eta_2 D2_t$$

The following statements estimate this GARCH model:

```
proc autoreg data=one;
   model y = x z / garch=(p=1,q=1);
   hetero d1 d2;
run;
```

The parameters for the variables D1 and D2 can be constrained using the COEF= option. For example, the constraints $\eta_1 = \eta_2 = 1$ are imposed by the following statements:

```
proc autoreg data=one;
   model y = x z / garch=(p=1,q=1);
   hetero d1 d2 / coef=unit;
run;
```

Limitations of GARCH and Heteroscedasticity Specifications

When you specify both the GARCH= option and the HETERO statement, the GARCH=(TYPE=EXP) option is not valid. The COVEST= option is not applicable to the EGARCH model.

EGARCH, IGARCH, GARCH-M Models

The AUTOREG procedure supports several variations of the generalized conditional heteroscedasticity model.

Using the TYPE= suboption of the GARCH= option, you can control the constraints placed on the estimated GARCH parameters. You can specify unconstrained, nonnegativity constrained (default), stationarity constrained, or integration constrained. The integration constraint produces the integrated GARCH or IGARCH model.

You can also use the TYPE= option to specify the exponential form of the GARCH model, called the EGARCH model. The MEAN suboption of the GARCH= option specifies the GARCH-in-mean or GARCH-M model.

The following statements illustrate the use of the TYPE= option to fit an AR(2)-EGARCH(1,1) model to the series Y. (Output is not shown.)

```
proc autoreg data=a;
   model y = time / nlag=2 garch=(p=1,q=1,type=exp);
run;
```

See the section “GARCH, IGARCH, EGARCH, and GARCH-M Models” later in this chapter for details.

Syntax

The AUTOREG procedure is controlled by the following statements:

PROC AUTOREG *options* **;**
BY *variables* **;**
MODEL *dependent = regressors / options* **;**
HETERO *variables / options* **;**
RESTRICT *equation , . . . , equation* **;**
TEST *equation , . . . , equation / option* **;**
OUTPUT *OUT = SAS data set options* **;**

At least one MODEL statement must be specified. One OUTPUT statement can follow each MODEL statement. One HETERO statement can follow each MODEL statement.

Functional Summary

The statements and options used with the AUTOREG procedure are summarized in the following table:

Description	Statement	Option
Data Set Options		
specify the input data set	AUTOREG	DATA=
write parameter estimates to an output data set	AUTOREG	OUTEST=
include covariances in the OUTEST= data set	AUTOREG	COVOUT
write predictions, residuals, and confidence limits to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
specify BY-group processing	BY	
Printing Control Options		
request all printing options	MODEL	ALL
print transformed coefficients	MODEL	COEF
print correlation matrix of the estimates	MODEL	CORRB
print covariance matrix of the estimates	MODEL	COVB
print DW statistics up to order j	MODEL	DW=j
print marginal probability of the generalized Durbin-Watson test statistics for large sample sizes	MODEL	DWPROB
print the *p*-values for the Durbin-Watson test be computed using a linearized approximation of the design matrix	MODEL	LDW

Description	Statement	Option
print inverse of Toeplitz matrix	MODEL	GINV
print the Godfrey LM serial correlation test	MODEL	GODFREY=
print details at each iteration step	MODEL	ITPRINT
print the Durbin *t* statistic	MODEL	LAGDEP
print the Durbin *h* statistic	MODEL	LAGDEP=
print the log likelihood value of the regression model	MODEL	LOGLIKL
print the Jarque-Bera normality test	MODEL	NORMAL
print tests for ARCH process	MODEL	ARCHTEST
print the Lagrange multiplier test	HETERO	TEST=LM
print the Chow test	MODEL	CHOW=
print the predictive Chow test	MODEL	PCHOW=
suppress printed output	MODEL	NOPRINT
print partial autocorrelations	MODEL	PARTIAL
print Ramsey's RESET test	MODEL	RESET
print tests for stationarity or unit roots	MODEL	STATIONARITY=(PHILLIPS=)
print tests of linear hypotheses	TEST	
specify the test statistics to use	TEST	TYPE=
prints the uncentered regression R^2	MODEL	URSQ
Model Estimation Options		
specify the order of autoregressive process	MODEL	NLAG=
center the dependent variable	MODEL	CENTER
suppress the intercept parameter	MODEL	NOINT
remove nonsignificant AR parameters	MODEL	BACKSTEP
specify significance level for BACKSTEP	MODEL	SLSTAY=
specify the convergence criterion	MODEL	CONVERGE=
specify the type of covariance matrix	MODEL	COVEST=
set the initial values of parameters used by the iterative optimization algorithm	MODEL	INITIAL=
specify iterative Yule-Walker method	MODEL	ITER
specify maximum number of iterations	MODEL	MAXITER=
specify the estimation method	MODEL	METHOD=
use only first sequence of nonmissing data	MODEL	NOMISS
specify the optimization technique	MODEL	OPTMETHOD=
imposes restrictions on the regression estimates	RESTRICT	
estimate and test heteroscedasticity models	HETERO	
GARCH Related Options		
specify order of GARCH process	MODEL	GARCH=(Q=,P=)
specify type of GARCH model	MODEL	GARCH=(. . .,TYPE=)
specify various forms of the GARCH-M model	MODEL	GARCH=(. . .,MEAN=)

Description	Statement	Option
suppress GARCH intercept parameter	MODEL	GARCH=(. . .,NOINT)
specify the trust region method	MODEL	GARCH=(. . .,TR)
estimate the GARCH model for the conditional *t*-distribution	MODEL	GARCH=(. . .) DIST=
estimates the start-up values for the conditional variance equation	MODEL	GARCH=(. . .,STARTUP=)
specify the functional form of the heteroscedasticity model	HETERO	LINK=
specify that the heteroscedasticity model does not include the unit term	HETERO	NOCONST
impose constraints on the estimated parameters the heteroscedasticity model	HETERO	COEF=
impose constraints on the estimated standard deviation of the heteroscedasticity model	HETERO	STD=
output conditional error variance	OUTPUT	CEV=
output conditional prediction error variance	OUTPUT	CPEV=
specify the flexible conditional variance form of the GARCH model	HETERO	
Output Control Options		
specify confidence limit size	OUTPUT	ALPHACLI=
specify confidence limit size for structural predicted values	OUTPUT	ALPHACLM=
specify the significance level for the upper and lower bounds of the CUSUM and CUSUMSQ statistics	OUTPUT	ALPHACSM=
specify the name of a variable to contain the values of the Theil's BLUS residuals	OUTPUT	BLUS=
output the value of the error variance σ_t^2	OUTPUT	CEV=
output transformed intercept variable	OUTPUT	CONSTANT=
specify the name of a variable to contain the CUSUM statistics	OUTPUT	CUSUM=
specify the name of a variable to contain the CUSUMSQ statistics	OUTPUT	CUSUMSQ=
specify the name of a variable to contain the upper confidence bound for the CUSUM statistic	OUTPUT	CUSUMUB=
specify the name of a variable to contain the lower confidence bound for the CUSUM statistic	OUTPUT	CUSUMLB=
specify the name of a variable to contain the upper confidence bound for the CUSUMSQ statistic	OUTPUT	CUSUMSQUB=

Description	Statement	Option
option specify the name of a variable to contain the lower confidence bound for the CUSUMSQ statistic	OUTPUT	CUSUMSQLB=
output lower confidence limit	OUTPUT	LCL=
output lower confidence limit for structural predicted values	OUTPUT	LCLM=
output predicted values	OUTPUT	P=
output predicted values of structural part	OUTPUT	PM=
output residuals	OUTPUT	R=
output residuals from structural predictions	OUTPUT	RM=
specify the name of a variable to contain the part of the predictive error variance (v_t)	OUTPUT	RECPEV=
specify the name of a variable to contain recursive residuals	OUTPUT	RECRES=
output transformed variables	OUTPUT	TRANSFORM=
output upper confidence limit	OUTPUT	UCL=
output upper confidence limit for structural predicted values	OUTPUT	UCLM=

PROC AUTOREG Statement

PROC AUTOREG *options ;*

The following options can be used in the PROC AUTOREG statement:

DATA= SAS-data-set

specifies the input SAS data set. If the DATA= option is not specified, PROC AUTOREG uses the most recently created SAS data set.

OUTEST= SAS-data-set

writes the parameter estimates to an output data set. See “OUTEST= Data Set” later in this chapter for information on the contents of this data set.

COVOUT

writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

In addition, any of the following MODEL statement options can be specified in the PROC AUTOREG statement, which is equivalent to specifying the option for every MODEL statement: ALL, ARCHTEST, BACKSTEP, CENTER, COEF, CONVERGE=, CORRB, COVB, DW=, DWPROB, GINV, ITER, ITPRINT, MAXITER=, METHOD=, NOINT, NOMISS, NOPRINT, and PARTIAL.

BY Statement

BY *variables;*

A BY statement can be used with PROC AUTOREG to obtain separate analyses on observations in groups defined by the BY variables.

MODEL Statement

MODEL *dependent = regressors / options ;*

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. If no independent variables are specified in the MODEL statement, only the mean is fitted. (This is a way to obtain autocorrelations of a series.)

Models can be given labels of up to eight characters. Model labels are used in the printed output to identify the results for different models. The model label is specified as follows:

label **: MODEL** . . . **;**

The following options can be used in the MODEL statement after a slash (/).

CENTER
centers the dependent variable by subtracting its mean and suppresses the intercept parameter from the model. This option is only valid when the model does not have regressors (explanatory variables).

NOINT
suppresses the intercept parameter.

Autoregressive Error Options

NLAG= *number*
NLAG= *(number-list)*
specifies the order of the autoregressive error process or the subset of autoregressive error lags to be fitted. Note that NLAG=3 is the same as NLAG=(1 2 3). If the NLAG= option is not specified, PROC AUTOREG does not fit an autoregressive model.

GARCH Estimation Options

DIST= *value*
specifies the distribution assumed for the error term in GARCH-type esimation. If no GARCH= option is specified, the option is ignored. If EGARCH is specified, the distribution is always normal distribution. The values of the DIST= option are

T	specifies Student's *t* distribution.
NORMAL	specifies the standard normal distribution. The default is DIST=NORMAL.

GARCH= *(option-list)*

Specifies a GARCH-type conditional heteroscedasticity model. The GARCH= option in the MODEL statement specifies the family of ARCH models to be estimated. The GARCH(1,1) regression model is specified in the following statement:

```
   model y = x1 x2 / garch=(q=1,p=1);
```

When you want to estimate the subset of ARCH terms, for example, ARCH(1 3), you can write the SAS statement as follows:

```
   model y = x1 x2 / garch=(q=(1 3));
```

With the TYPE= option, you can specify various GARCH models. The IGARCH(2,1) model without trend in variance is estimated as follows:

```
   model y = / garch=(q=2,p=1,type=integ,noint);
```

The following options can be used in the GARCH=( ) option. The options are listed within parentheses and separated by commas.

Q= *number*
Q= *(number-list)*

specifies the order of the process or the subset of ARCH terms to be fitted.

P= *number*
P= *(number-list)*

specifies the order of the process or the subset of GARCH terms to be fitted. If only the P= option is specified, Q=1 is assumed.

TYPE= *value*

specifies the type of GARCH model. The values of the TYPE= option are

EXP	specifies the exponential GARCH or EGARCH model.
INTEGRATED	specifies the integrated GARCH or IGARCH model.
NELSON | NELSONCAO	specifies the Nelson-Cao inequality constraints.
NONNEG	specifies the GARCH model with nonnegativity constraints.
STATIONARY	constrains the sum of GARCH coefficients to be less than 1.

The default is TYPE=NELSON.

MEAN= *value*

specifies the functional form of the GARCH-M model. The values of the MEAN= option are

LINEAR specifies the linear function.

$$y_t = \mathbf{x}_t^{'}\beta + \delta h_t + \epsilon_t$$

LOG specifies the log function.

$$y_t = \mathbf{x}_t^{'}\beta + \delta \mathrm{ln h_t} + \mathrm{\epsilon_t}$$

SQRT specifies the square root function.

$$y_t = \mathbf{x}_t^{'}\beta + \delta\sqrt{h_t} + \epsilon_t$$

NOINT

suppresses the intercept parameter in the conditional variance model. This option is valid only with the TYPE=INTEG option.

STARTUP= MSE | ESTIMATE

STARTUP=ESTIMATE requests that the positive constant *c* for the start-up values of the GARCH conditional error variance process be estimated. By default or if STARTUP=MSE is specified, the value of the mean squared error is used as the default constant.

TR

uses the trust region method for GARCH estimation. This algorithm is numerically stable, though computation is expensive. The double quasi-Newton method is the default.

Printing Options

ALL

requests all printing options.

ARCHTEST

requests the Q and LM statistics testing for the absence of ARCH effects.

CHOW= *(obs*$_1$ *... obs*$_n$ *)*

The CHOW= option computes Chow tests to evaluate the stability of the regression coefficient. The Chow test is also called the analysis-of-variance test.

Each value obs_i listed on the CHOW= option specifies a break point of the sample. The sample is divided into parts at the specified break point, with observations before obs_i in the first part and obs_i and later observations in the second part, and the fits of the model in the two parts are compared to whether both parts of the sample are consistent with the same model.

The break points obs_i refer to observations within the time range of the dependent variable, ignoring missing values before the start of the dependent series. Thus, CHOW=20 specifies the twentieth observation after the first nonmissing observation for the dependent variable. For example, if the dependent variable Y contains 10 missing values before the first observation with a nonmissing Y value, then CHOW=20 actually refers to the 30th observation in the data set.

When you specify the break point, you should note the number of pre-sample missing values.

COEF
prints the transformation coefficients for the first p observations. These coefficients are formed from a scalar multiplied by the inverse of the Cholesky root of the Toeplitz matrix of autocovariances.

CORRB
prints the estimated correlations of the parameter estimates.

COVB
prints the estimated covariances of the parameter estimates.

COVEST= OP | HESSIAN | QML
The COVEST= option specifies the type of covariance matrix for the GARCH or heteroscedasticity model. When COVEST=OP is specified, the outer product matrix is used to compute the covariance matrix of the parameter estimates. The COVEST=HESSIAN option produces the covariance matrix using the Hessian matrix. The quasi-maximum likelihood estimates are computed with COVEST=QML. The default is COVEST=OP.

DW= *n*
prints Durbin-Watson statistics up to the order *n*. The default is DW=1. When the LAGDEP option is specified, the Durbin-Watson statistic is not printed unless the DW= option is explicitly specified.

DWPROB
The DWPROB option now produces *p*-values for the generalized Durbin-Watson test statistics for large sample sizes. Previously, the Durbin-Watson probabilities were calculated only for small sample sizes. The new method of calculating Durbin-Watson probabilities is based on the algorithm of Ansley, Kohn, and Shively (1992).

GINV
prints the inverse of the Toeplitz matrix of autocovariances for the Yule-Walker solution. See “Computational Methods” later in this chapter for details.

GODFREY
GODFREY= *r*
The GODFREY option produces Godfrey’s general Lagrange multiplier test against ARMA errors.

ITPRINT
prints the objective function and parameter estimates at each iteration. The objective function is the full log likelihood function for the maximum likelihood method, while the error sum of squares is produced as the objective function of unconditional least squares. For the ML method, the ITPRINT option prints the value of the full log likelihood function, not the concentrated likelihood.

LAGDEP
LAGDV

prints the Durbin *t* statistic, which is used to detect residual autocorrelation in the presence of lagged dependent variables. See "Generalized Durbin-Watson Tests" later in this chapter for details.

LAGDEP= *name*
LAGDV= *name*

prints the Durbin *h* statistic for testing the presence of first-order autocorrelation when regressors contain the lagged dependent variable whose name is specified as LAGDEP=*name*. If the Durbin *h* statistic cannot be computed, the asymptotically equivalent *t* statistic is printed instead. See "Generalized Durbin-Watson Tests" for details.

When the regression model contains several lags of the dependent variable, specify the lagged dependent variable for the smallest lag in the LAGDEP= option, for example,

```
model y = x1 x2 ylag2 ylag3 / lagdep=ylag2;
```

LOGLIKL

The LOGLIKL option prints the log likelihood value of the regression model, assuming normally distributed errors.

NOPRINT

suppresses all printed output.

NORMAL

The NORMAL option specifies the Jarque-Bera's normality test statistic for regression residuals.

PARTIAL

prints partial autocorrelations.

PCHOW= *(obs*$_1$ *... obs*$_n$ *)*

The PCHOW= option computes the predictive Chow test. The form of the PCHOW= option is the same as the CHOW= option; see the discussion of the CHOW= option earlier in this chapter.

RESET

The RESET option produces Ramsey's RESET test statistics. The RESET option tests the null model

$$y_t = \mathbf{x}_t\beta + u_t$$

against the alternative

$$y_t = \mathbf{x}_t\beta + \sum_{j=2}^{p} \phi_j \hat{y}_t^j + u_t$$

where $\hat{y}_t$ is the predicted value from the OLS estimation of the null model. The RESET option produces three RESET test statistics for $p = 2$, 3, and 4.

STATIONARITY= *(PHILLIPS)*
STATIONARITY= *(PHILLIPS=(value ... value))*

The STATIONARITY= option specifies tests of stationarity or unit roots. The STATIONARITY= option provides Phillips-Perron tests.

The PHILLIPS or PHILLIPS= suboption of the STATIONARITY= option produces the Phillips-Perron unit root test when there are no regressors in the MODEL statement. When the model includes regressors, the PHILLIPS option produces the Phillips-Ouliaris cointegration test. The PHILLIPS option can be abbreviated as PP.

The PHILLIPS option performs the Phillips-Perron test for three null hypothesis cases: zero mean, single mean, and deterministic trend. For each case, the PHILLIPS option computes two test statistics, Z($\hat{\alpha}$) and $\mathrm{Z}(\mathrm{t}_{\hat{\alpha}})$, and reports their *p*-values. These test statistics have the same limiting distributions as the corresponding Dickey-Fuller tests.

The three types of the Phillips-Perron unit root test reported by the PHILLIPS option are as follows.

Zero Mean	computes the Phillips-Perron test statistic based on the zero mean autoregressive model $$y_t = \alpha y_{t-1} + u_t$$
Single Mean	computes the Phillips-Perron test statistic based on the autoregressive model with a constant term $$y_t = \mu + \alpha y_{t-1} + u_t$$
Trend	computes the Phillips-Perron test statistic based on the autoregressive model with constant and time trend terms $$y_t = \mu + \alpha y_{t-1} + \delta t + u_t$$

You can specify several truncation points l for weighted variance estimators using the PHILLIPS=($l_1 \ldots l_n$) specification. The statistic for each truncation point l is computed as

$$\sigma^2_{Tl} = \frac{1}{T}\sum_{i=1}^{T}\hat{u}_i^2 + \frac{2}{T}\sum_{s=1}^{l} w_{sl}\sum_{t=s+1}^{T}\hat{u}_t\hat{u}_{t-s}$$

where $w_{sl} = 1 - s/(l+1)$ and $\hat{u}_t$ are OLS residuals. If you specify the PHILLIPS option without specifying truncation points, the default truncation point is $\max(1, \sqrt{T}/5)$, where T is the number of observations.

The Phillips-Perron test can be used in general time series models since its limiting distribution is derived in the context of a class of weakly dependent and heterogeneously distributed data. The marginal probability for the Phillips-Perron test is computed assuming that error disturbances are normally distributed.

When there are regressors in the MODEL statement, the PHILLIPS option computes the cointegration test statistic using the least squares residuals. The normalized cointegrating vector is estimated using OLS regression. Therefore, the cointegrating vector estimates might vary with regressand (normalized element) unless the regression R-square is 1.

The marginal probabilities for cointegration testing are not produced. You can refer to Phillips and Ouliaris (1990) tables Ia-Ic for the $Z(\hat{\alpha})$ test and tables IIa-IIc for the $Z(t_{\hat{\alpha}})$ test. The standard residual-based cointegration test can be obtained using the NOINT option in the MODEL statement, while the demeaned test is computed by including the intercept term. To obtain the demeaned and detrended cointegration tests, you should include the time trend variable in the regressors. Refer to Phillips and Ouliaris (1990) or Hamilton (1994) for information about the Phillips-Ouliaris cointegration test.

URSQ

The URSQ option prints the uncentered regression R^2. The uncentered regression R^2 is useful to compute Lagrange multiplier test statistics, since most LM test statistics are computed as *T**URSQ, where *T* is the number of observations used in estimation.

Stepwise Selection Options

BACKSTEP

removes insignificant autoregressive parameters. The parameters are removed in order of least significance. This backward elimination is done only once on the Yule-Walker estimates computed after the initial ordinary least-squares estimation. The BACKSTEP option can be used with all estimation methods since the initial parameter values for other estimation methods are estimated using the Yule-Walker method.

SLSTAY= *value*

specifies the significance level criterion to be used by the BACKSTEP option. The default is SLSTAY=.05.

Estimation Control Options

CONVERGE= *value*

specifies the convergence criterion. If the maximum absolute value of the change in the autoregressive parameter estimates between iterations is less than this amount, then convergence is assumed. The default is CONVERGE=.001.

INITIAL= *(initial-values)*
START= *(initial-values)*

The INITIAL= option specifies initial values for some or all of the parameter estimates. The values specified are assigned to model parameters in the same order as the parameter estimates are printed in the AUTOREG procedure output. The order of values in the INITIAL= or START= option is: the intercept, the regressor coefficients, the autoregressive parameters, the ARCH parameters, the GARCH parameters, the inverted degrees of freedom for Student's *t* distribution, the start-up value for conditional variance, and the heteroscedasticity model parameters η specified by the HETERO statement.

The following is an example of specifying initial values for an AR(1)-GARCH(1,1) model with regressors X1 and X2:

```
model y = w x / nlag=1 garch=(p=1,q=1)
                initial=(1 1 1 .5 .8 .1 .6);
```

The model specified by this MODEL statement is

$$y_t = \beta_0 + \beta_1 w_t + \beta_2 x_t + \nu_t$$

$$\nu_t = \epsilon_t - \phi_1 \nu_{t-1}$$

$$\epsilon_t = \sqrt{h_t} e_t$$

$$h_t = \omega + \alpha_1 \epsilon_{t-1}^2 + \gamma_1 h_{t-1}$$

$$\epsilon_t \ \mathrm{N}(0, \sigma_{\mathrm{t}}^2)$$

The initial values for the regression parameters, INTERCEPT (β_0), X1 (β_1), and X2 (β_2), are specified as 1. The initial value of the AR(1) coefficient (ϕ_1) is specified as 0.5. The initial value of ARCH0 (ω) is 0.8, the initial value of ARCH1 (α_1) is 0.1, and the initial value of GARCH1 (γ_1) is 0.6.

When you use the RESTRICT statement, the initial values specified by the INITIAL= option should satisfy the restrictions specified for the parameter estimates. If they do not, the initial values you specify are adjusted to satisfy the restrictions.

LDW

The LDW option specifies that *p*-values for the Durbin-Watson test be computed using a linearized approximation of the design matrix when the model is nonlinear due to the presence of an autoregressive error process. (The Durbin-Watson tests of the OLS linear regression model residuals are not affected by the LDW option.) Refer to White (1992) for Durbin-Watson testing of nonlinear models.

MAXITER= *number*

sets the maximum number of iterations allowed. The default is MAXITER=50.

METHOD= *value*

requests the type of estimates to be computed. The values of the METHOD= option are

METHOD=ML	specifies maximum likelihood estimates
METHOD=ULS	specifies unconditional least-squares estimates
METHOD=YW	specifies Yule-Walker estimates
METHOD=ITYW	specifies iterative Yule-Walker estimates

If the GARCH= or LAGDEP option is specified, the default is METHOD=ML. Otherwise, the default is METHOD=YW.

NOMISS

requests the estimation to the first contiguous sequence of data with no missing values. Otherwise, all complete observations are used.

OPTMETHOD= QN | TR

The OPTMETHOD= option specifies the optimization technique when the GARCH or heteroscedasticity model is estimated. The OPTMETHOD=QN option specifies the quasi-Newton method. The OPTMETHOD=TR option specifies the trust region method. The default is OPTMETHOD=QN.

HETERO Statement

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way these variables are used to model the error variance of the regression.

The syntax of the HETERO statement is

HETERO *variables / options* **;**

The heteroscedastic regression model supported by the HETERO statement is

$$y_t = \mathbf{x}_t\beta + \epsilon_t$$

$$\epsilon_t \ \mathrm{N}(0, \sigma_{\mathrm{t}}^2)$$

$$\sigma_t^2 = \sigma^2 h_t$$

$$h_t = l(\mathbf{z}_t^{'}\eta)$$

The HETERO statement specifies a model for the conditional variance h_t. The vector $\mathbf{z}_t$ is composed of the variables listed on the HETERO statement, η is a parameter vector, and $l(\cdot)$ is a link function that depends on the value of the LINK= option.

The keyword XBETA can be used in the *variables* list to refer to the model predicted value $\mathbf{x}_t^{'}\beta$. If XBETA is specified in the *variables* list, other variables in the HETERO statement will be ignored. In addition, XBETA cannot be specified in the GARCH process.

The errors ϵ_t are assumed to be uncorrelated– the heteroscedasticity models specified by the HETERO statement cannot be combined with an autoregressive model for the errors. Thus, the HETERO statement cannot be used if the NLAG= option is specified in the MODEL statement.

You can specify the following options in the HETERO statement:

LINK= *value*

The LINK= option specifies the functional form of the heteroscedasticity model. If you want to estimate the GARCH model whose conditional error variance contains exogenous variables, you do not need to specify the LINK= option. The default is LINK=EXP. Values of the LINK= option are

EXP specifies the exponential link function. The following model is estimated when you specify LINK=EXP:

$$h_t = \exp(\mathbf{z}_t^{'}\eta)$$

SQUARE specifies the square link function. The following model is estimated when you specify LINK=SQUARE:

$$h_t = (1 + \mathbf{z}_t^{'}\eta)^2$$

LINEAR specifies the linear function; that is, the HETERO statement variables predict the error variance linearly. The following model is estimated when you specify LINK=LINEAR:

$$h_t = (1 + \mathbf{z}_t^{'}\eta)$$

COEF= *value*

The COEF= option imposes constraints on the estimated parameters η of the heteroscedasticity model. The values of the COEF= option are

NONNEG specifies that the estimated heteroscedasticity parameters η must be nonnegative. When the HETERO statement is used in conjunction with the GARCH= option, the default is COEF=NONNEG.

UNIT	constrains all heteroscedasticity parameters η to equal 1.
ZERO	constrains all heteroscedasticity parameters η to equal 0.
UNREST	specifies unrestricted estimation of η. When the GARCH= option is not specified, the default is COEF=UNREST.

STD= *value*

The STD= option imposes constraints on the estimated standard deviation σ of the heteroscedasticity model. The values of the STD= option are

NONNEG	specifies that the estimated standard deviation parameter σ must be nonnegative.
UNIT	constrains the standard deviation parameter σ to equal 1.
UNREST	specifies unrestricted estimation of σ. This is the default.

TEST= *LM*

The TEST=LM option produces a Lagrange multiplier test for heteroscedasticity. The null hypothesis is homoscedasticity; the alternative hypothesis is heteroscedasticity of the form specified by the HETERO statement. The power of the test depends on the variables specified in the HETERO statement.

The test may give different results depending on the functional form specified by the LINK= option. However, in many cases the test does not depend on the LINK= option. The test is invariant to the form of h_t when $h_t(0) = 1$ and $h_t'(0) ne 0$. (The condition $h_t(0) = 1$ is satisfied except when the NOCONST option is specified with LINK=SQUARE or LINK=LINEAR.)

NOCONST

The NOCONST option specifies that the heteroscedasticity model does not include the unit term for the LINK=SQUARE and LINK=LINEAR options. For example, the following model is estimated when you specify the options LINK=SQUARE NOCONST:

$$h_t = (\mathbf{z}_t'\eta)^2$$

RESTRICT Statement

The RESTRICT statement provides constrained estimation.

RESTRICT *equation , ... , equation ;*

The RESTRICT statement places restrictions on the parameter estimates for covariates in the preceding MODEL statement. Any number of RESTRICT statements can follow a MODEL statement. Several restrictions can be specified in a single RESTRICT statement by separating the individual restrictions with commas.

Each restriction is written as a linear equation composed of constants and parameter names. Refer to model parameters by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model.

The following is an example of a RESTRICT statement:

```
model y = a b c d;
restrict a+b=0, 2*d-c=0;
```

When restricting a linear combination of parameters to be 0, you can omit the equal sign. For example, the following RESTRICT statement is equivalent to the preceding example:

```
restrict a+b, 2*d-c;
```

The following RESTRICT statement constrains the parameters estimates for three regressors (X1, X2, and X3) to be equal:

```
restrict x1 = x2, x2 = x3;
```

The preceding restriction can be abbreviated as follows.

```
restrict x1 = x2 = x3;
```

Only simple linear combinations of parameters can be specified in RESTRICT statement expressions; complex expressions involving parentheses, division, functions, or complex products are not allowed.

TEST Statement

The AUTOREG procedure now supports a TEST statement for linear hypothesis tests.

TEST *equation* , ... , *equation* / *option* **;**

The TEST statement tests hypotheses about the covariates in the model estimated by the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. If more than one equation is specified, the equations are separated by commas.

Each test is written as a linear equation composed of constants and parameter names. Refer to parameters by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model.

You can specify the following option in the TEST statement:

TYPE= *value*

The TYPE= option specifies the test statistics to use, *F* or Wald. TYPE=F produces an *F*-test. TYPE=WALD produces a Wald test. The default is TYPE=F.

The following example of a TEST statement tests the hypothesis that the coefficients of two regressors A and B are equal:

```
model y = a b c d;
test a = b;
```

To test separate null hypotheses, use separate TEST statements. To test a joint hypothesis, specify the component hypotheses on the same TEST statement, separated by commas.

For example, consider the following linear model:

$$y_t = \beta_0 + \beta_1 x1_t + \beta_2 x2_t + \epsilon_t$$

The following statements test the two hypotheses $H_0 : \beta_0 = 1$ and $H_0 : \beta_1 + \beta_2 = 0$:

```
model y = x1 x2;
test intercept = 1;
test x1 + x2 = 0;
```

The following statements test the joint hypothesis $H_0 : \beta_0 = 1$ and $\beta_1 + \beta_2 = 0$:

```
model y = x1 x2;
test intercept = 1, x1 + x2 = 0;
```

OUTPUT Statement

OUTPUT OUT= *SAS-data-set keyword = options . . .;*

The OUTPUT statement creates an output SAS data set as specified by the following options:

OUT= *SAS-data-set*

names the output SAS data set containing the predicted and transformed values. If the OUT= option is not specified, the new data set is named according to the DATA*n* convention.

ALPHACLI= *number*

sets the confidence limit size for the estimates of future values of the response time series. The ALPHACLI= value must be between 0 and 1. The resulting confidence interval has 1-*number* confidence. The default is ALPHACLI=.05, corresponding to a 95% confidence interval.

ALPHACLM= *number*

sets the confidence limit size for the estimates of the structural or regression part of the model. The ALPHACLI= value must be between 0 and 1. The resulting confidence interval has 1-*number* confidence. The default is ALPHACLM=.05, corresponding to a 95% confidence interval.

ALPHACSM= .01 | .05 | .10

The ALPHACSM= option specifies the significance level for the upper and lower bounds of the CUSUM and CUSUMSQ statistics output by the CUSUMLB=, CUSUMUB=, CUSUMSQLB=, and CUSUMSQUB= options. The significance level specified by the ALPHACSM= option can be .01, .05, or .10. Other values are not supported.

The following options are of the form *KEYWORD=name*, where *KEYWORD* specifies the statistic to include in the output data set and *name* gives the name of the variable in the OUT= data set containing the statistic.

BLUS= *variable*

The BLUS= option specifies the name of a variable to contain the values of the Theil's BLUS residuals. Refer to Theil (1971) for more information on BLUS residuals.

CEV= *variable*
HT= *variable*

The CEV= option writes to the output data set the value of the error variance σ_t^2 from the heteroscedasticity model specified by the HETERO statement or the value of the conditional error variance h_t by the GARCH= option in the MODEL statement.

CPEV= *variable*

writes the conditional prediction error variance to the output data set. The value of conditional prediction error variance is equal to that of the conditional error variance when there are no autoregressive parameters. For the exponential GARCH model, conditional prediction error variance cannot be calculated. See "Predicted Values" later in this chapter for details.

CONSTANT= *variable*

writes the transformed intercept to the output data set. The details of the transformation are described in "Computational Methods" later in this chapter.

CUSUM= *variable*

The CUSUM= option specifies the name of a variable to contain the CUSUM statistics.

CUSUMSQ= *variable*

The CUSUMSQ= option specifies the name of a variable to contain the CUSUMSQ statistics.

CUSUMUB= *variable*

The CUSUMUB= option specifies the name of a variable to contain the upper confidence bound for the CUSUM statistic.

CUSUMLB= *variable*
The CUSUMLB= option specifies the name of a variable to contain the lower confidence bound for the CUSUM statistic.

CUSUMSQUB= *variable*
The CUSUMSQUB= option specifies the name of a variable to contain the upper confidence bound for the CUSUMSQ statistic.

CUSUMSQLB= *variable*
The CUSUMSQLB= option specifies the name of a variable to contain the lower confidence bound for the CUSUMSQ statistic.

LCL= *name*
writes the lower confidence limit for the predicted value (specified in the PREDICTED= option) to the output data set. The size of the confidence interval is set by the ALPHACLI= option. When a GARCH model is estimated, the lower confidence limit is calculated assuming that the disturbances have homoscedastic conditional variance. See "Predicted Values" later in this chapter for details.

LCLM= *name*
writes the lower confidence limit for the structural predicted value (specified in the PREDICTEDM= option) to the output data set under the name given. The size of the confidence interval is set by the ALPHACLM= option.

PREDICTED= *name*
P= *name*
writes the predicted values to the output data set. These values are formed from both the structural and autoregressive parts of the model. See "Predicted Values" later in this chapter for details.

PREDICTEDM= *name*
PM= *name*
writes the structural predicted values to the output data set. These values are formed from only the structural part of the model. See "Predicted Values" later in this chapter for details.

RECPEV= *variable*
The RECPEV= option specifies the name of a variable to contain the part of the predictive error variance (v_t) that is used to compute the recursive residuals.

RECRES= *variable*
The RECRES= option specifies the name of a variable to contain recursive residuals. The recursive residuals are used to compute the CUSUM and CUSUMSQ statistics.

RESIDUAL= *name*
R= *name*
writes the residuals from the predicted values based on both the structural and time series parts of the model to the output data set.

RESIDUALM= *name*
RM= *name*
writes the residuals from the structural prediction to the output data set.

TRANSFORM= *variables*
transforms the specified variables from the input data set by the autoregressive model and writes the transformed variables to the output data set. The details of the transformation are described in “Computational Methods” later in this chapter. If you need to reproduce the data suitable for reestimation, you must also transform an intercept variable. To do this, transform a variable that is all 1s or use the CONSTANT= option.

UCL= *name*
writes the upper confidence limit for the predicted value (specified in the PREDICTED= option) to the output data set. The size of the confidence interval is set by the ALPHACLI= option. When the GARCH model is estimated, the upper confidence limit is calculated assuming that the disturbances have homoscedastic conditional variance. See “Predicted Values” later in this chapter for details.

UCLM= *name*
writes the upper confidence limit for the structural predicted value (specified in the PREDICTEDM= option) to the output data set. The size of the confidence interval is set by the ALPHACLM= option.

Details

Missing Values

PROC AUTOREG skips any missing values at the beginning of the data set. If the NOMISS option is specified, the first contiguous set of data with no missing values is used; otherwise, all data with nonmissing values for the independent and dependent variables are used. Note, however, that the observations containing missing values are still needed to maintain the correct spacing in the time series. PROC AUTOREG can generate predicted values when the dependent variable is missing.

Autoregressive Error Model

The regression model with autocorrelated disturbances is as follows:

$$y_t = \mathbf{x}_t'\beta + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1\nu_{t-1} - \ldots - \varphi_m\nu_{t-m}$$

$$\epsilon_t \; \mathrm{N}(0, \sigma^2)$$

In these equations, y_t are the dependent values, $\mathbf{x}_t$ is a column vector of regressor variables, β is a column vector of structural parameters, and ϵ_t is normally and independently distributed with a mean of 0 and a variance of σ^2. Note that in this parameterization, the signs of the autoregressive parameters are reversed from the parameterization documented in most of the literature.

PROC AUTOREG offers four estimation methods for the autoregressive error model. The default method, Yule-Walker (YW) estimation, is the fastest computationally. The Yule-Walker method used by PROC AUTOREG is described in Gallant and Goebel (1976). Harvey (1981) calls this method the *two-step full transform method*. The other methods are iterated YW, unconditional least squares (ULS), and maximum likelihood (ML). The ULS method is also referred to as nonlinear least squares (NLS) or exact least squares (ELS).

You can use all of the methods with data containing missing values, but you should use ML estimation if the missing values are plentiful. See the section "Alternative Autocorrelation Correction Methods" later in this chapter for further discussion of the advantages of different methods.

The Yule-Walker Method

Let φ represent the vector of autoregressive parameters

$$\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_m)'$$

and let the variance matrix of the error vector $\nu = (\nu_1, \ldots, \nu_N)'$ be Σ

$$E(\nu\nu') = \mathbf{\Sigma} = \sigma^2\mathbf{V}$$

If the vector of autoregressive parameters φ is known, the matrix **V** can be computed from the autoregressive parameters. Σ is then $\sigma^2\mathbf{V}$. Given Σ, the efficient estimates of regression parameters β can be computed using generalized least squares (GLS). The GLS estimates then yield the unbiased estimate of the variance σ^2,

The Yule-Walker method alternates estimation of β using generalized least squares with estimation of φ using the Yule-Walker equations applied to the sample autocorrelation function. The YW method starts by forming the OLS estimate of β. Next, φ is estimated from the sample autocorrelation function of the OLS residuals using the Yule-Walker equations. Then **V** is estimated from the estimate of φ, and Σ is estimated from **V** and the OLS estimate of σ^2. The autocorrelation corrected estimates of the regression parameters β are then computed by GLS using the estimated Σ matrix. These are the Yule-Walker estimates.

If the ITER option is specified, the Yule-Walker residuals are used to form a new sample autocorrelation function, the new autocorrelation function is used to form a new estimate of φ and **V**, and the GLS estimates are recomputed using the new variance matrix. This alternation of estimates continues until either the maximum change in the $\widehat{\varphi}$ estimate between iterations is less than the value specified by the CONVERGE= option or the maximum number of allowed iterations is reached. This produces the Iterated Yule-Walker estimates. Iteration of the estimates may not yield much improvement.

The Yule-Walker equations, solved to obtain $\widehat{\varphi}$ and a preliminary estimate of σ^2, are

$$\mathbf{R}\hat{\varphi} = -\mathbf{r}$$

Here $\mathbf{r} = (r_1, \ldots, r_m)'$, where r_i is the lag *i* sample autocorrelation. The matrix **R** is the Toeplitz matrix whose *i,j*th element is $r_{|i-j|}$. If you specify a subset model, then only the rows and columns of **R** and **r** corresponding to the subset of lags specified are used.

If the BACKSTEP option is specified, for purposes of significance testing, the matrix $[\mathbf{R}\ \mathbf{r}]$ is treated as a sum-of-squares-and-crossproducts matrix arising from a simple regression with $N - k$ observations, where *k* is the number of estimated parameters.

The Unconditional Least Squares and Maximum Likelihood Methods

Define the transformed error, $\mathbf{e}$, as

$$\mathbf{e} = \mathbf{L}^{-1}\mathbf{n}$$

where $\mathbf{n} = \mathbf{y} - \mathbf{X}\beta$.

The unconditional sum of squares for the model, S, is

$$S = \mathbf{n}'\mathbf{V}^{-1}\mathbf{n} = \mathbf{e}'\mathbf{e}$$

The ULS estimates are computed by minimizing S with respect to the parameters β and φ_i.

The full log likelihood function for the autoregressive error model is

$$l = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|\mathbf{V}|) - \frac{S}{2\sigma^2}$$

where $|\mathbf{V}|$ denotes determinant of $\mathbf{V}$. For the ML method, the likelihood function is maximized by minimizing an equivalent sum-of-squares function.

Maximizing *l* with respect to σ^2 (and concentrating σ^2 out of the likelihood) and dropping the constant term $-\frac{N}{2}\ln(2\pi) + 1 - \ln(N)$ produces the concentrated log likelihood function

$$l_c = -\frac{N}{2}\ln(S|\mathbf{V}|^{1/N})$$

Rewriting the variable term within the logarithm gives

$$S_{ml} = |\mathbf{L}|^{1/N}\mathbf{e}'\mathbf{e}|\mathbf{L}|^{1/N}$$

PROC AUTOREG computes the ML estimates by minimizing the objective function $S_{ml} = |\mathbf{L}|^{1/N}\mathbf{e}'\mathbf{e}|\mathbf{L}|^{1/N}$.

The maximum likelihood estimates may not exist for some data sets (Anderson and Mentz 1980). This is the case for very regular data sets, such as an exact linear trend.

Computational Methods

Sample Autocorrelation Function

The sample autocorrelation function is computed from the structural residuals or noise $\mathbf{n_t} = y_t - \mathbf{x}_t'\mathbf{b}$, where $\mathbf{b}$ is the current estimate of β. The sample autocorrelation function is the sum of all available lagged products of $\mathbf{n}_t$ of order *j* divided by $\ell + j$, where ℓ is the number of such products.

If there are no missing values, then $\ell + j = N$, the number of observations. In this case, the Toeplitz matrix of autocorrelations, **R**, is at least positive semidefinite. If there are missing values, these autocorrelation estimates of *r* can yield an **R** matrix that is not positive semidefinite. If such estimates occur, a warning message is printed, and the estimates are tapered by exponentially declining weights until **R** is positive definite.

Data Transformation and the Kalman Filter

The calculation of **V** from φ for the general AR(*m*) model is complicated, and the size of **V** depends on the number of observations. Instead of actually calculating **V** and performing GLS in the usual way, in practice a Kalman filter algorithm is used to transform the data and compute the GLS results through a recursive process.

In all of the estimation methods, the original data are transformed by the inverse of the Cholesky root of **V**. Let **L** denote the Cholesky root of **V**, that is $\mathbf{V} = \mathbf{LL}'$ with **L** lower triangular. For an AR(*m*) model, $\mathbf{L}^{-1}$ is a band diagonal matrix with *m* anomalous rows at the beginning and the autoregressive parameters along the remaining rows. Thus, if there are no missing values, after the first *m*-1 observations the data are transformed as

$$z_t = x_t + \hat{\varphi}_1 x_{t-1} + \ldots + \hat{\varphi}_m x_{t-m}$$

The transformation is carried out using a Kalman filter, and the lower triangular matrix **L** is never directly computed. The Kalman filter algorithm, as it applies here, is described in Harvey and Phillips (1979) and Jones (1980). Although **L** is not computed explicitly, for ease of presentation the remaining discussion is in terms of **L**. If there are missing values, then the submatrix of **L** consisting of the rows and columns with nonmissing values is used to generate the transformations.

Gauss-Newton Algorithms

The ULS and ML estimates employ a Gauss-Newton algorithm to minimize the sum of squares and maximize the log likelihood, respectively. The relevant optimization is performed simultaneously for both the regression and AR parameters. The OLS estimates of β and the Yule-Walker estimates of φ are used as starting values for these methods.

The Gauss-Newton algorithm requires the derivatives of **e** or $|\mathbf{L}|^{1/N}\mathbf{e}$ with respect to the parameters. The derivatives with respect to the parameter vector β are

$$\frac{\partial \mathbf{e}}{\partial \beta'} = -\mathbf{L}^{-1}\mathbf{X}$$

$$\frac{\partial|\mathbf{L}|^{1/N}\mathbf{e}}{\partial\beta'} = -|\mathbf{L}|^{1/N}\mathbf{L}^{-1}\mathbf{X}$$

These derivatives are computed by the transformation described previously. The derivatives with respect to φ are computed by differentiating the Kalman filter recurrences and the equations for the initial conditions.

Variance Estimates and Standard Errors

For the Yule-Walker method, the estimate of the error variance, s^2, is the error sum of squares from the last application of GLS, divided by the error degrees of freedom (number of observations *N* minus the number of free parameters).

The variance-covariance matrix for the components of **b** is taken as $s^2(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$ for the Yule-Walker method. For the ULS and ML methods, the variance-covariance matrix of the parameter estimates is computed as $s^2(\mathbf{J}'\mathbf{J})^{-1}$. For the ULS method, **J** is the matrix of derivatives of **e** with respect to the parameters. For the ML method, **J** is the matrix of derivatives of $|\mathbf{L}|^{1/N}\mathbf{e}$ divided by $|\mathbf{L}|^{1/N}$. The estimate of the variance-covariance matrix of **b** assuming that φ is known is $s^2(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$.

Park and Mitchell (1980) investigated the small sample performance of the standard error estimates obtained from some of these methods. In particular, simulating an AR(1) model for the noise term, they found that the standard errors calculated using GLS with an estimated autoregressive parameter underestimated the true standard errors. These estimates of standard errors are the ones calculated by PROC AUTOREG with the Yule-Walker method.

The estimates of the standard errors calculated with the ULS or ML methods take into account the joint estimation of the AR and the regression parameters and may give more accurate standard-error values than the YW method. At the same values of the autoregressive parameters, the ULS and ML standard errors will always be larger than those computed from Yule-Walker. However, simulations of the models used by Park and Mitchell suggest that the ULS and ML standard error estimates can also be underestimates. Caution is advised, especially when the estimated autocorrelation is high and the sample size is small.

High autocorrelation in the residuals is a symptom of lack of fit. An autoregressive error model should not be used as a nostrum for models that simply do not fit. It is often the case that time series variables tend to move as a random walk. This means that an AR(1) process with a parameter near one absorbs a great deal of the variation. See Example 12.3 later in this chapter, which fits a linear trend to a sine wave.

For ULS or ML estimation, the joint variance-covariance matrix of all the regression and autoregression parameters is computed. For the Yule-Walker method, the variance-covariance matrix is computed only for the regression parameters.

Lagged Dependent Variables

The Yule-Walker estimation method is not directly appropriate for estimating models that include lagged dependent variables among the regressors. Therefore, the maximum likelihood method is the default when the LAGDEP or LAGDEP= option is specified in the MODEL statement. However, when lagged dependent variables are used, the maximum likelihood estimator is not exact maximum likelihood but is conditional on the first few values of the dependent variable.

Alternative Autocorrelation Correction Methods

Autocorrelation correction in regression analysis has a long history, and various approaches have been suggested. Moreover, the same method may be referred to by different names.

Pioneering work in the field was done by Cochrane and Orcutt (1949). The *Cochrane-Orcutt method* refers to a more primitive version of the Yule-Walker method that drops the first observation. The Cochrane-Orcutt method is like the Yule-Walker method for first-order autoregression, except that the Yule-Walker method retains information from the first observation. The iterative Cochrane-Orcutt method is also in use.

The Yule-Walker method used by PROC AUTOREG is also known by other names. Harvey (1981) refers to the Yule-Walker method as the *two-step full transform method*. The Yule-Walker method can be considered as generalized least squares using the OLS residuals to estimate the covariances across observations, and Judge et al. (1985) use the term *estimated generalized least squares* (EGLS) for this method. For a first-order AR process, the Yule-Walker estimates are often termed *Prais-Winsten estimates* (Prais and Winsten 1954). There are variations to these methods that use different estimators of the autocorrelations or the autoregressive parameters.

The unconditional least squares (ULS) method, which minimizes the error sum of squares for all observations, is referred to as the nonlinear least squares (NLS) method by Spitzer (1979).

The *Hildreth-Lu* method (Hildreth and Lu 1960) uses nonlinear least squares to jointly estimate the parameters with an AR(1) model, but it omits the first transformed residual from the sum of squares. Thus, the Hildreth-Lu method is a more primitive version of the ULS method supported by PROC AUTOREG in the same way Cochrane-Orcutt is a more primitive version of Yule-Walker.

The maximum likelihood method is also widely cited in the literature. Although the maximum likelihood method is well defined, some early literature refers to estimators that are called maximum likelihood but are not full unconditional maximum likelihood estimates. The AUTOREG procedure produces full unconditional maximum likelihood estimates.

Harvey (1981) and Judge et al. (1985) summarize the literature on various estimators for the autoregressive error model. Although asymptotically efficient, the various methods have different small sample properties. Several Monte Carlo experiments have been conducted, although usually for the AR(1) model.

Harvey and McAvinchey (1978) found that for a one-variable model, when the independent variable is trending, methods similar to Cochrane-Orcutt are inefficient in estimating the structural parameter. This is not surprising since a pure trend model is well modeled by an autoregressive process with a parameter close to 1.

Harvey and McAvinchey (1978) also made the following conclusions:

- The Yule-Walker method appears to be about as efficient as the maximum likelihood method. Although Spitzer (1979) recommended ML and NLS, the Yule-Walker method (labeled Prais-Winsten) did as well or better in estimating the structural parameter in Spitzer's Monte Carlo study (table A2 in their article) when the autoregressive parameter was not too large. Maximum likelihood tends to do better when the autoregressive parameter is large.
- For small samples, it is important to use a full transformation (Yule-Walker) rather than the Cochrane-Orcutt method, which loses the first observation. This was also demonstrated by Maeshiro (1976), Chipman (1979), and Park and Mitchell (1980).
- For large samples (Harvey used 100), losing the first few observations does not make much difference.

GARCH, IGARCH, EGARCH, and GARCH-M Models

Consider the series y_t, which follows the GARCH process. The conditional distribution of the series Y for time *t* is written

$$y_t|\Psi_{t-1}{\sim}\mathrm{N}(0,\mathrm{h_t})$$

where Ψ_{t-1} denotes all available information at time $t-1$. The conditional variance h_t is

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i y_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

where

$$p{\geq}0, q > 0$$

$$\omega > 0, \alpha_i{\geq}0, \gamma_j{\geq}0$$

The GARCH(p,q) model reduces to the ARCH(q) process when $p = 0$. At least one of the ARCH parameters must be nonzero ($q > 0$). The GARCH regression model can be written

$$y_t = \mathbf{x}_t'\beta + \epsilon_t$$

$$\epsilon_t = \sqrt{h_t}e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

where $e_t \sim \mathrm{IN}(0,1)$.

In addition, you can consider the model with disturbances following an autoregressive process and with the GARCH errors. The AR(m)-GARCH(p,q) regression model is denoted

$$y_t = \mathbf{x}_t' \beta + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1} - \ldots - \varphi_m \nu_{t-m}$$

$$\epsilon_t = \sqrt{h_t} e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

GARCH Estimation with Nelson-Cao Inequality Constraints

The GARCH(p,q) model is written in ARCH(∞) form as

$$\begin{aligned} h_t &= \left(1 - \sum_{j=1}^{p} \gamma_j B^j\right)^{-1} \left[\omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2\right] \\ &= \omega^* + \sum_{i=1}^{\infty} \phi_i \epsilon_{t-i}^2 \end{aligned}$$

where *B* is a backshift operator. Therefore, $h_t \geq 0$ if $\omega^* \geq 0$ and $\phi_i \geq 0, \forall i$. Assume that the roots of the following polynomial equation are inside the unit circle:

$$\sum_{j=0}^{p} -\gamma_j Z^{p-j}$$

where $\gamma_0 = -1$ and *Z* is a complex scalar. $\sum_{j=0}^{p} -\gamma_j Z^{p-j}$ and $\sum_{i=1}^{q} \alpha_i Z^{q-i}$, do not share common factors. Under these conditions, $|\omega^*| < \infty$, $|\phi_i| < \infty$, and these coefficients of the ARCH(∞) process are well defined.

Define $n = max(p,q)$. The coefficient ϕ_i is written

$$\phi_0 = \alpha_1$$

$$\begin{aligned}\phi_1 &= \gamma_1\phi_0 + \alpha_2 \\ \cdots & \\ \phi_{n-1} &= \gamma_1\phi_{n-2} + \gamma_2\phi_{n-3} + \cdots + \gamma_{n-1}\phi_0 + \alpha_n \\ \phi_k &= \gamma_1\phi_{k-1} + \gamma_2\phi_{k-2} + \cdots + \gamma_n\phi_{k-n} \quad \text{for}\ \ \text{k}\geq\text{n}\end{aligned}$$

where $\alpha_i = 0$ for $i > q$ and $\gamma_j = 0$ for $j > p$.

Nelson and Cao (1992) proposed the finite inequality constraints for GARCH(1,q) and GARCH(2,q) cases. However, it is not straightforward to derive the finite inequality constraints for the general GARCH(p,q) model.

For the GARCH(1,q) model, the nonlinear inequality constraints are

$$\begin{aligned}\omega &\geq 0 \\ \gamma_1 &\geq 0 \\ \phi_k &\geq 0 \quad \text{for}\ \ \text{k} = 0, 1, \cdots, \text{q} - 1\end{aligned}$$

For the GARCH(2,q) model, the nonlinear inequality constraints are

$$\begin{aligned}\Delta_i &\in R \quad \text{for}\ \ \text{i} = 1, 2 \\ \omega^* &\geq 0 \\ \Delta_1 &> 0 \\ \sum_{j=0}^{q-1} \Delta_1^{-j}\alpha_{j+1} &> 0 \\ \phi_k &\geq 0 \quad \text{for}\ \ \text{k} = 0, 1, \cdots, \text{q}\end{aligned}$$

where Δ_1 and Δ_2 are the roots of $(Z^2 - \gamma_1 Z - \gamma_2)$.

For the GARCH(p,q) model with $p > 2$, only max(q-1,p)+1 nonlinear inequality constraints ($\phi_k \geq 0$ for $k = 0$ to max$(q-1, p)$) are imposed, together with the in-sample positivity constraints of the conditional variance h_t.

Using the HETERO Statement with GARCH Models

The HETERO statement can be combined with the GARCH= option on the MODEL statement to include input variables in the GARCH conditional variance model. For example, the GARCH(1,1) variance model with two dummy input variables D1 and D2 is

$$\begin{aligned}\epsilon_t &= \sqrt{h_t}e_t \\ h_t &= \omega + \alpha_1\epsilon_{t-1}^2 + \gamma_1 h_{t-1} + \eta_1 DI_t + \eta_2 DI_t\end{aligned}$$

The following statements estimate this GARCH model:

```
   proc autoreg data=one;
      model y = x z / garch=(p=1,q=1);
      hetero d1 d2;
   run;
```

The parameters for the variables D1 and D2 can be constrained using the COEF= option. For example, the constraints $\eta_1 = \eta_2 = 1$ are imposed by the following statements:

```
   proc autoreg data=one;
      model y = x z / garch=(p=1,q=1);
      hetero d1 d2 / coef=unit;
   run;
```

Limitations of GARCH and Heteroscedasticity Specifications

When you specify both the GARCH= option and the HETERO statement, the GARCH=(TYPE=EXP) option is not valid. The COVEST= option is not applicable to the EGARCH model.

IGARCH and Stationary GARCH Model

The condition $\sum_{i=1}^{q}\alpha_i + \sum_{j=1}^{p}\gamma_j < 1$ implies that

the GARCH process is weakly stationary since the mean, variance, and autocovariance are finite and constant over time. However, this condition is not sufficient for weak stationarity in the presence of autocorrelation. For example, the stationarity condition for an AR(1)-GARCH(p,q) process is

$$\frac{1}{1-\varphi_1^2}\sum_{i=1}^{q}\alpha_i + \sum_{j=1}^{p}\gamma_j < 1$$

When the GARCH process is stationary, the unconditional variance of ϵ_t is computed as

$$\mathbf{V}(\epsilon_t) = \frac{\omega}{(1-\sum_{i=1}^{q}\alpha_i - \sum_{j=1}^{p}\gamma_j)}$$

where $\epsilon_t = \sqrt{h_t}e_t$ and h_t is the

GARCH(*p*,*q*) conditional variance.

Sometimes, the multistep forecasts of the variance do not approach the unconditional variance when the model is integrated in variance; that is, $\sum_{i=1}^{q}\alpha_i + \sum_{j=1}^{p}\gamma_j = 1$.

The unconditional variance for the IGARCH model does not exist. However, it is interesting that the IGARCH model can be strongly stationary even though it is not weakly stationary. Refer to Nelson (1990) for details.

EGARCH Model

The EGARCH model was proposed by Nelson (1991). Nelson and Cao (1992) argue that the nonnegativity constraints in the linear GARCH model are too restrictive. The GARCH model imposes the nonnegative constraints on the parameters, α_i and γ_j, while there are no restrictions on these parameters in the EGARCH model. In the EGARCH model, the conditional variance, h_t, is an asymmetric function of lagged disturbances ϵ_{t-i}:

$$\ln(h_t) = \omega + \sum_{i=1}^{q} \alpha_i g(z_{t-i}) + \sum_{j=1}^{p} \gamma_j \ln(h_{t-j})$$

where

$$g(z_t) = \theta z_t + \gamma[|z_t| - E|z_t|]$$

$$z_t = \epsilon_t / \sqrt{h_t}$$

The coefficient of the second term in $g(z_t)$ is set to be 1 (γ=1) in our formulation. Note that $E|z_t| = (2/\pi)^{1/2}$ if $z_t{\sim}\mathrm{N}(0,1)$. The properties of the EGARCH model are summarized as follows:

- The function $g(z_t)$ is linear in z_t with slope coefficient $\theta + 1$ if z_t is positive while $g(z_t)$ is linear in z_t with slope coefficient $\theta - 1$ if z_t is negative
- Suppose that $\theta = 0$. Large innovations increase the conditional variance if $|z_t| - E|z_t| > 0$ and decrease
- the conditional variance if $|z_t| - E|z_t| < 0$.
- Suppose that $\theta < 1$. The innovation in variance, $g(z_t)$, is positive if the innovations z_t are less than $(2/\pi)^{1/2}/(\theta - 1)$. Therefore, the negative innovations in returns, ϵ_t, cause the innovation to the conditional variance to be positive if θ is much less than 1.

GARCH-in-Mean

The GARCH-M model has the added regressor that is the conditional standard deviation:

$$y_t = \mathbf{x}_t'\beta + \delta\sqrt{h_t} + \epsilon_t$$

$$\epsilon_t = \sqrt{h_t} e_t$$

where h_t follows the ARCH or GARCH process.

Maximum Likelihood Estimation

The family of GARCH models are estimated using the maximum likelihood method. The log-likelihood function is computed from the product of all conditional densities of the prediction errors.

When e_t is assumed to have a standard normal distribution ($e_t \sim \mathrm{N}(0,1)$), the likelihood function is given by

$$l = \sum_{t=1}^{N} \frac{1}{2} \left[-\ln(2\pi) - \ln(h_t) - \frac{\epsilon_t^2}{h_t} \right]$$

where $\epsilon_t = y_t - \mathbf{x}_t'\beta$ and h_t is the conditional variance. When the GARCH(p,q)-M model is estimated, $\epsilon_t = y_t - \mathbf{x}_t'\beta - \delta\sqrt{h_t}$. When there are no regressors, the residuals ϵ_t are denoted as y_t or $y_t - \delta\sqrt{h_t}$.

If e_t has the standardized Student's *t* distribution the log likelihood function for the conditional *t* distribution is

$$\ell = \sum_{t=1}^{N} \left[\log\left(\Gamma\left(\frac{\nu+1}{2}\right)\right) - \log\left(\Gamma\left(\frac{\nu}{2}\right)\right) - \frac{1}{2}\log((\nu-2)h_t) \right.$$
$$\left. -\frac{1}{2}(\nu+1)\log\left(1 + \frac{\epsilon_t^2}{h_t(\nu-2)}\right) \right]$$

where $\Gamma(\cdot)$ is the gamma function and ν is the degree of freedom ($\nu > 2$). Under the conditional *t* distribution, the additional parameter $1/\nu$ is estimated. The log likelihood function for the conditional *t* distribution converges to the log likelihood function of the conditional normal GARCH model as $1/\nu \rightarrow 0$.

The likelihood function is maximized via either the dual quasi-Newton or trust region algorithm. The default is the dual quasi-Newton algorithm. The starting values for the regression parameters β are obtained from the OLS estimates. When there are autoregressive parameters in the model, the initial values are obtained from the Yule-Walker estimates. The starting value 1.0^{-6} is used for the GARCH process parameters.

The variance-covariance matrix is computed using the Hessian matrix. The dual quasi-Newton method approximates the Hessian matrix while the quasi-Newton method gets an approximation of the inverse of Hessian. The trust region method uses the Hessian matrix obtained using numerical differentiation. When there are active constraints, that is, $\mathbf{q}(\theta) = \mathbf{0}$, the variance-covariance matrix is given by

$$\mathbf{V}(\hat{\theta}) = \mathbf{H}^{-1}[\mathbf{I} - \mathbf{Q}'(\mathbf{Q}\mathbf{H}^{-1}\mathbf{Q}')^{-1}\mathbf{Q}\mathbf{H}^{-1}]$$

where $\mathbf{H} = -\partial^2 l/\partial\theta\partial\theta'$ and $\mathbf{Q} = \partial\mathbf{q}(\theta)/\partial\theta'$. Therefore, the variance-covariance matrix without active constraints reduces to $\mathbf{V}(\hat{\theta}) = \mathbf{H}^{-1}$.

R^2 Statistics and Other Measures of Fit

This section discusses various goodness-of-fit statistics produced by the AUTOREG procedure.

Total R^2

The total R^2 statistic (Total Rsq) is computed as

$$R^2_{\mathrm{tot}} = 1 - \frac{SSE}{SST}$$

where *SST* is the sum of squares for the original response variable corrected for the mean and SSE is the final error sum of squares. The Total Rsq is a measure of how well the next value can be predicted using the structural part of the model and the past values of the residuals. If the NOINT option is specified, *SST* is the uncorrected sum of squares.

Regression R^2

The regression R^2 (Reg RSQ) is computed as

$$R^2_{\mathrm{reg}} = 1 - \frac{TSSE}{TSST}$$

where *TSST* is the total sum of squares of the transformed response variable corrected for the transformed intercept, and *TSSE* is the error sum of squares for this transformed regression problem. If the NOINT option is requested, no correction for the transformed intercept is made. The Reg RSQ is a measure of the fit of the structural part of the model after transforming for the autocorrelation and is the R^{2}for the transformed regression.

The regression R^2 and the total R^2 should be the same when there is no autocorrelation correction (OLS regression).

Calculation of Recursive Residuals and CUSUM Statistics

The recursive residuals w_t are computed as

$$w_t = \frac{e_t}{\sqrt{v_t}}$$

$$v_t = 1 + \mathbf{x}_t^{'} \left[\sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{x}_i^{'} \right]^{-1} \mathbf{x}_t$$

Note that the forecast error variance of e_t is the scalar multiple of v_t such that $V(e_t) = \sigma^2 v_t$.

The CUSUM and CUSUMSQ statistics are computed using the preceding recursive residuals.

$$\mathrm{CUSUM_t} = \sum_{\mathrm{i=k+1}}^{\mathrm{t}} \frac{\mathrm{w_i}}{\sigma_{\mathrm{w}}}$$

$$\mathrm{CUSUMSQ_t} = \frac{\sum_{\mathrm{i=k+1}}^{\mathrm{t}} \mathrm{w_i^2}}{\sum_{\mathrm{i=k+1}}^{\mathrm{T}} \mathrm{w_i^2}}$$

where w_i are the recursive residuals,

$$\sigma_w = \sqrt{\frac{\sum_{i=k+1}^{T} (w_i - \hat{w})^2}{(T-k-1)}}$$

$$\hat{w} = \frac{1}{T-k} \sum_{i=k+1}^{T} w_i$$

and k is the number of regressors.

The CUSUM statistics can be used to test for misspecification of the model. The upper and lower critical values for CUSUM$_t$ are

$$\pm a \left[\sqrt{T-k} + 2 \frac{(t-k)}{(T-k)^{\frac{1}{2}}} \right]$$

where *a* = 1.143 for a significance level .01, 0.948 for .05, and 0.850 for .10. These critical values are output by the CUSUMLB= and CUSUMUB= options for the significance level specified by the ALPHACSM= option.

The upper and lower critical values of CUSUMSQ$_t$ are given by

$$\pm a + \frac{(t-k)}{T-k}$$

where the value of *a* is obtained from the table by Durbin (1969) if the $\frac{1}{2}(T-k)-1 \leq 60$. Edgerton and Wells (1994) provided the method of obtaining the value of *a* for large samples.

These critical values are output by the CUSUMSQLB= and CUSUMSQUB= options for the significance level specified by the ALPHACSM= option.

Information Criteria AIC and SBC

The Akaike's information criterion (AIC) and the Schwarz's Bayesian information criterion (SBC) are computed as follows:

$$\mathrm{AIC} = -2\mathrm{ln}(\mathrm{L}) + 2\mathrm{k}$$

$$\mathrm{SBC} = -2\mathrm{ln}(\mathrm{L}) + \mathrm{ln}(\mathrm{N})\mathrm{k}$$

In these formulas, *L* is the value of the likelihood function evaluated at the parameter estimates, *N* is the number of observations, and *k* is the number of estimated parameters. Refer to Judge et al. (1985) and Schwarz (1978) for additional details.

Generalized Durbin-Watson Tests

Consider the following linear regression model:

$$\mathbf{Y} = \mathbf{X}\beta + \nu$$

where **X** is an $N*k$ data matrix, β is a $k*1$ coefficient vector, and ν is a $N*1$ disturbance vector. The error term ν is assumed to be generated by the *j*th order autoregressive process ν_t=ϵ_t-$\varphi_j\nu_{t-j}$ where $|\varphi_j| < 1$, ϵ_t is a sequence of independent normal error terms with mean 0 and variance σ^2. Usually, the Durbin-Watson statistic is used to test the null hypothesis $H_0 : \varphi_1 = 0$ against $H_1 : -\varphi_1 > 0$. Vinod (1973) generalized the Durbin-Watson statistic:

$$d_j = \frac{\sum_{t=j+1}^{N} (\hat{\nu}_t - \hat{\nu}_{t-j})^2}{\sum_{t=1}^{N} \hat{\nu}_t^2}$$

where $\hat{\nu}$ are OLS residuals. Using the matrix notation,

$$d_j = \frac{\mathbf{Y'MA'_jA_jMY}}{\mathbf{Y'MY}}$$

where $\mathbf{M} = \mathbf{I}_N - \mathbf{X}(\mathbf{X'X})^{-1}\mathbf{X'}$ and $\mathbf{A}_j$ is a $(N-j)\times N$ matrix:

$$\mathbf{A}_j = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and there are $j-1$ zeros between -1 and 1 in each row of matrix $\mathbf{A}_j$.

The QR factorization of the design matrix **X** yields a $N*N$ orthogonal matrix **Q**

$$\mathbf{X} = \mathbf{QR}$$

where R is a $N * k$ upper triangular matrix. There exists a $N * (N-k)$ submatrix of **Q** such that $\mathbf{Q}_1\mathbf{Q}_1' = \mathbf{M}$ and $\mathbf{Q}_1'\mathbf{Q}_1 = \mathbf{I}_{N-k}$. Consequently, the generalized Durbin-Watson statistic is stated as a ratio of two quadratic forms:

$$d_j = \frac{\sum_{l=1}^{n} \lambda_{jl}{\xi_l}^2}{\sum_{l=1}^{n} \xi_l^2}$$

where $\lambda_{j1} \ldots \lambda_{jn}$ are upper *n* eigenvalues of $\mathbf{M}\mathbf{A}_j^{'}\mathbf{A}_j\mathbf{M}$ and ξ_l is a standard normal variate, and $n = \min(N-k, N-j)$. These eigenvalues are obtained by a singular value decomposition of $\mathbf{Q}_1^{'}\mathbf{A}_j^{'}$ (Golub and Loan 1989; Savin and White 1978).

The marginal probability (or *p*-value) for d_j given c_0 is

$$\mathrm{Prob}(\frac{\sum_{\mathrm{l=1}}^{\mathrm{n}} \lambda_{\mathrm{jl}}\xi_{\mathrm{l}}^2}{\sum_{\mathrm{l=1}}^{\mathrm{n}} \xi_{\mathrm{l}}^2} < \mathrm{c_0}) = \mathrm{Prob}(\mathrm{q_j} < 0)$$

where

$$q_j = \sum_{l=1}^{n} (\lambda_{jl} - c_0)\xi_l^2$$

When the null hypothesis $H_0 : \varphi_j = 0$ holds, the quadratic form q_j has the characteristic function

$$\phi_j(t) = \prod_{l=1}^{n} (1 - 2(\lambda_{jl} - c_0)it)^{-1/2}$$

The distribution function is uniquely determined by this characteristic function:

$$F(x) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\infty \frac{e^{itx}\phi_j(-t) - e^{-itx}\phi_j(t)}{it} dt$$

For example, to test $H_0 : \varphi_4 = 0$ given $\varphi_1 = \varphi_2 = \varphi_3 = 0$ against $H_1 : -\varphi_4 > 0$, the marginal probability (*p*-value) can be used:

$$F(0) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\infty \frac{(\phi_4(-t) - \phi_4(t))}{it} dt$$

where

$$\phi_4(t) = \prod_{l=1}^{n} (1 - 2(\lambda_{4l} - \hat{d}_4)it)^{-1/2}$$

and $\hat{d}_4$ is the calculated value of the fourth-order Durbin-Watson statistic.

In the Durbin-Watson test, the marginal probability indicates positive autocorrelation ($-\varphi_j > 0$) if it is less than the level of significance (α), while you can conclude that a negative autocorrelation ($-\varphi_j < 0$) exists if the marginal probability based on the computed Durbin-Watson statistic is greater than 1-α. Wallis (1972) presented tables for bounds tests of fourth-order autocorrelation and Vinod (1973) has given tables for a five percent significance level for orders two to four. Using the AUTOREG procedure, you can calculate the exact *p*-values for the general order of Durbin-Watson test statistics. Tests for the absence of autocorrelation of order *p* can be performed sequentially; at the *j*th step, test $H_0 : \varphi_j = 0$ given $\varphi_1 = \ldots = \varphi_{j-1} = 0$ against $\varphi_j \neq 0$. However, the size of the sequential test is not known.

The Durbin-Watson statistic is computed from the OLS residuals, while that of the autoregressive error model uses residuals that are the difference between the predicted values and the actual values. When you use the Durbin-Watson test from the residuals of the autoregressive error model, you must be aware that this test is only an approximation. See “Regression with Autoregressive Errors” earlier in this chapter. If there are missing values, the Durbin-Watson statistic is computed using all the nonmissing values and ignoring the gaps caused by missing residuals. This does not affect the significance level of the resulting test, although the power of the test against certain alternatives may be adversely affected. Savin and White (1978) have examined the use of the Durbin-Watson statistic with missing values.

Enhanced Durbin-Watson Probability Computation

The Durbin-Watson probability calculations have been enhanced to compute the p-value of the generalized Durbin-Watson statistic for large sample sizes. Previously, the Durbin-Watson probabilities were only calculated for small sample sizes.

Consider the following linear regression model:

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{u}$$

$$u_t + \varphi_j u_{t-j} = \epsilon_t, \qquad t = 1, \ldots, N$$

where $\mathbf{X}$ is an $N \times k$ data matrix, β is a $k \times 1$ coefficient vector, $\mathbf{u}$ is a $N \times 1$ disturbance vector, ϵ_t is a sequence of independent normal error terms with mean 0 and variance σ^2.

The generalized Durbin-Watson statistic is written as

$$\mathrm{DW}_j = \frac{\hat{\mathbf{u}}'\mathbf{A}_j'\mathbf{A}_j\hat{\mathbf{u}}}{\hat{\mathbf{u}}'\hat{\mathbf{u}}}$$

where $\hat{\mathbf{u}}$ is a vector of OLS residuals and $\mathbf{A}_j$ is a $(T-j)\times T$ matrix. The generalized Durbin-Watson statistic DW$_j$ can be rewritten as

$$\mathrm{DW}_j = \frac{\mathbf{Y}'\mathbf{M}\mathbf{A}_j'\mathbf{A}_j\mathbf{M}\mathbf{Y}}{\mathbf{Y}'\mathbf{M}\mathbf{Y}} = \frac{\eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1)\eta}{\eta'\eta}$$

where $\mathbf{Q}_1'\mathbf{Q}_1 = \mathbf{I}_{T-k},\ \mathbf{Q}_1'\mathbf{X} = 0$, and $\eta = \mathbf{Q}_1'\mathbf{u}$.

The marginal probability for the Durbin-Watson statistic is

$$\Pr(\mathrm{DW}_j < c) = \Pr(h < 0)$$

where $h = \eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I})\eta$.

The p-value or the marginal probability for the generalized Durbin-Watson statistic is computed by numerical inversion of the characteristic function $\phi(u)$ of the quadratic form $h = \eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I})\eta$. The trapezoidal rule approximation to the marginal probability $\Pr(h < 0)$ is

$$\Pr(h<0) = \frac{1}{2} - \sum_{k=0}^{K} \frac{\mathrm{Im}\left[\phi((k+\frac{1}{2})\Delta)\right]}{\pi(k+\frac{1}{2})} + \mathrm{E}_I(\Delta) + \mathrm{E}_T(K)$$

where $\mathrm{Im}\left[\phi(\cdot)\right]$ is the imaginary part of the characteristic function, $\mathrm{E}_I(\Delta)$ and $\mathrm{E}_T(K)$ are integration and truncation errors, respectively. Refer to Davies (1973) for numerical inversion of the characteristic function.

Ansley, Kohn, and Shively (1992) proposed a numerically efficient algorithm which requires O(N) operations for evaluation of the characteristic function $\phi(u)$. The characteristic function is denoted as

$$\begin{aligned} \phi(u) &= \left|\mathbf{I} - 2iu(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I}_{N-k})\right|^{-1/2} \\ &= |\mathbf{V}|^{-1/2}\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{-1/2}\left|\mathbf{X}'\mathbf{X}\right|^{1/2} \end{aligned}$$

where $\mathbf{V} = (1 + 2iuc)\mathbf{I} - 2iu\mathbf{A}_j'\mathbf{A}_j$ and $i = \sqrt{-1}$. By applying the Cholesky decomposition to the complex matrix $\mathbf{V}$, you can obtain the lower triangular matrix $\mathbf{G}$ which satisfies $\mathbf{V} = \mathbf{G}\mathbf{G}'$. Therefore, the characteristic function can be evaluated in O(N) operations using the following formula:

$$\phi(u) = |\mathbf{G}|^{-1}\left|\mathbf{X}^{*\prime}\mathbf{X}^*\right|^{-1/2}\left|\mathbf{X}'\mathbf{X}\right|^{1/2}$$

where $\mathbf{X}^* = \mathbf{G}^{-1}\mathbf{X}$. Refer to Ansley, Kohn, and Shively (1992) for more information on evaluation of the characteristic function.

Tests for Serial Correlation with Lagged Dependent Variables

When regressors contain lagged dependent variables, the Durbin-Watson statistic (d_1) for the first-order autocorrelation is biased toward 2 and has reduced power. Wallis (1972) shows that the bias in the Durbin-Watson statistic (d_4) for the fourth-order autocorrelation is smaller than the bias in d_1 in the presence of a first-order lagged dependent variable. Durbin (1970) proposed two alternative statistics (Durbin *h* and *t*) that are asymptotically equivalent. The *h* statistic is written as

$$h = \hat{\rho}\sqrt{N/(1 - N\hat{V})}$$

where $\hat{\rho} = \sum_{t=2}^{N} \hat{\nu}_t \hat{\nu}_{t-1} / \sum_{t=1}^{N} \hat{\nu}_t^2$ and $\hat{V}$ is the least-squares variance estimate for the coefficient of the lagged dependent variable. Durbin's *t*-test consists of regressing the OLS residuals $\hat{\nu}_t$ on explanatory variables and $\hat{\nu}_{t-1}$ and testing the significance of the estimate for coefficient of $\hat{\nu}_{t-1}$.

Inder (1984) shows that the Durbin-Watson test for the absence of first-order autocorrelation is generally more powerful than the *h*-test in finite samples. Refer to Inder (1986) and King and Wu (1991) for the Durbin-Watson test in the presence of lagged dependent variables.

Testing

Heteroscedasticity and Normality Tests

Portmanteau Q-Test

For nonlinear time series models, the portmanteau test statistic based on squared residuals is used to test for independence of the series (McLeod and Li 1983):

$$Q(q) = N(N+2)\sum_{i=1}^{q} \frac{r(i; \hat{\nu}_t^2)}{(N-i)}$$

where

$$r(i; \hat{\nu}_t^2) = \frac{\sum_{t=i+1}^{N} (\hat{\nu}_t^2 - \hat{\sigma}^2)(\hat{\nu}_{t-i}^2 - \hat{\sigma}^2)}{\sum_{t=1}^{N} (\hat{\nu}_t^2 - \hat{\sigma}^2)^2}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{t=1}^{N} \hat{\nu}_t^2$$

This *Q* statistic is used to test the nonlinear effects (for example, GARCH effects) present in the residuals. The GARCH(p,q) process can be considered as an ARMA(max(p,q),p) process. See the section "Predicting the Conditional Variance" later in this chapter. Therefore, the *Q* statistic calculated from the squared residuals can be used to identify the order of the GARCH process.

Lagrange Multiplier Test for ARCH Disturbances

Engle (1982) proposed a Lagrange multiplier test for ARCH disturbances. The test statistic is asymptotically equivalent to the test used by Breusch and Pagan (1979). Engle's Lagrange multiplier test for the *q*th order ARCH process is written

$$LM(q) = \frac{N\mathbf{W}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{W}}{\mathbf{W}'\mathbf{W}}$$

where

$$\mathbf{W} = \left(\frac{\hat{\nu}_1^2}{\hat{\sigma}^2}, \ldots, \frac{\hat{\nu}_N^2}{\hat{\sigma}^2}\right)'$$

and

$$\mathbf{Z} = \begin{bmatrix} 1 & \hat{\nu}_0^2 & \cdots & \hat{\nu}_{-q+1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \hat{\nu}_{N-1}^2 & \cdots & \hat{\nu}_{N-q}^2 \end{bmatrix}$$

The presample values (ν_0^2,. . ., ν_{-q+1}^2) have been set to 0. Note that the LM(q) tests may have different finite sample properties depending on the presample values, though they are asymptotically equivalent regardless of the presample values. The LM and *Q* statistics are computed from the OLS residuals assuming that disturbances are white noise. The *Q* and LM statistics have an approximate $\chi^2_{(q)}$ distribution under the white-noise null hypothesis.

Normality Test

Based on skewness and kurtosis, Bera and Jarque (1982) calculated the test statistic

$$T_N = \left[\frac{N}{6}b_1^2 + \frac{N}{24}(b_2 - 3)^2\right]$$

where

$$b_1 = \frac{\sqrt{N}\sum_{t=1}^{N}\hat{u}_t^3}{\left(\sum_{t=1}^{N}\hat{u}_t^2\right)^{\frac{3}{2}}}$$

$$b_2 = \frac{N\sum_{t=1}^{N}\hat{u}_t^4}{\left(\sum_{t=1}^{N}\hat{u}_t^2\right)^{2}}$$

The χ^2(2)-distribution gives an approximation to the normality test T_N.

When the GARCH model is estimated, the normality test is obtained using the standardized residuals $\hat{u}_t = \hat{\epsilon}_t/\sqrt{h_t}$. The normality test can be used to detect misspecification of the family of ARCH models.

Computation of the Chow Test

Consider the linear regression model

$$\mathbf{y} = X\beta + \mathbf{u}$$

where the parameter vector β contains k elements.

Split the observations for this model into two subsets at the break point specified by the CHOW= option, so that $\mathbf{y} = (\mathbf{y}_1^{'}, \mathbf{y}_2^{'})^{'}$,

$X = (X_1^{'}, X_2^{'})^{'}$, and

$\mathbf{u} = (\mathbf{u}_1^{'}, \mathbf{u}_2^{'})^{'}$.

Now consider the two linear regressions for the two subsets of the data modeled separately,

$$\mathbf{y}_1 = X_1\beta_1 + \mathbf{u}_1$$

$$\mathbf{y}_2 = X_2\beta_2 + \mathbf{u}_2$$

where the number of observations from the first set is n_1 and the number of observations from the second set is n_2.

The Chow test statistic is used to test the null hypothesis $H_0 : \beta_1 = \beta_2$ conditional on the same error variance $V(\mathbf{u}_1) = V(\mathbf{u}_2)$. The Chow test is computed using three sums of square errors.

$$\mathrm{F}_{chow} = \frac{(\hat{\mathbf{u}}'\hat{\mathbf{u}} - \hat{\mathbf{u}}_1^{'}\hat{\mathbf{u}}_1 - \hat{\mathbf{u}}_2^{'}\hat{\mathbf{u}}_2)/\mathrm{k}}{(\hat{\mathbf{u}}_1^{'}\hat{\mathbf{u}}_1 + \hat{\mathbf{u}}_2^{'}\hat{\mathbf{u}}_2)/(\mathrm{n}_1 + \mathrm{n}_2 - 2\mathrm{k})}$$

where $\hat{\mathbf{u}}$ is the regression residual vector from the full set model, $\hat{\mathbf{u}}_1$ is the regression residual vector from the first set model, and $\hat{\mathbf{u}}_2$ is the regression residual vector from the second set model. Under the null hypothesis, the Chow test statistic has an *F*-distribution with k and $(n_1 + n_2 - 2k)$ degrees of freedom, where k is the number of elements in β.

Chow (1960) suggested another test statistic that tests the hypothesis that the mean of prediction errors is 0. The predictive Chow test can also be used when $n_2 < k$.

The PCHOW= option computes the predictive Chow test statistic

$$\mathrm{F}_{pchow} = \frac{(\hat{\mathbf{u}}'\hat{\mathbf{u}} - \hat{\mathbf{u}}_1^{'}\hat{\mathbf{u}}_1)/\mathrm{n}_2}{\hat{\mathbf{u}}_1^{'}\hat{\mathbf{u}}_1/(\mathrm{n}_1 + \mathrm{k})}$$

The predictive Chow test has an *F*-distribution with n_2 and $(n_1 - k)$ degrees of freedom.

Unit Root and Cointegration Testing

Consider the random walk process

$$y_t = y_{t-1} + u_t$$

where the disturbances might be serially correlated with possible heteroscedasticity. Phillips and Perron (1988) proposed the unit root test of the OLS regression model.

$$y_t = \alpha y_{t-1} + u_t$$

Let $s^2 = \frac{1}{T-k}\sum_{t=1}^{T}\hat{u}_t^2$ and let $\hat{\sigma}^2$ be the variance estimate of the OLS estimator $\hat{\alpha}$, where $\hat{u}_t$ is the OLS residual. You can estimate the asymptotic variance of $\frac{1}{T}\sum_{t=1}^{T}\hat{u}_t^2$using the truncation lag *l*.

$$\hat{\lambda} = \sum_{j=0}^{l}\kappa_j[1 - j/(l+1)]\hat{\gamma}_j$$

where $\kappa_0 = 1$, $\kappa_j = 2$ for $j > 0$, and $\hat{\gamma}_j = \frac{1}{T}\sum_{t=j+1}^{T}\hat{u}_t\hat{u}_{t-j}$.

Then the Phillips-Perron Z($\hat{\alpha}$) test (zero mean case) is written

$$\mathrm{Z}(\hat{\alpha}) = T(\hat{\alpha} - 1) - \frac{1}{2}T^2\hat{\sigma}^2(\hat{\lambda} - \hat{\gamma}_0)/\mathrm{s}^2$$

and has the following limiting distribution:

$$\frac{\frac{1}{2}\{B(1)^2 - 1\}}{\int_0^1 [B(x)]^2 dx}$$

where *B*(·) is a standard Brownian motion. Note that the realization Z(x) from the the stochastic process B(·) is distributed as N(0,x) and thus $B(1)^2 \sim \chi_1^2$.

Therefore, you can observe that $\mathrm{P}(\hat{\alpha} < 1) \approx 0.68$ as $T \to \infty$, which shows that the limiting distribution is skewed to the left.

Let $t_{\hat{\alpha}}$ be the *t*-test statistic for $\hat{\alpha}$. The Phillips-Perron $\mathrm{Z}(\mathrm{t}_{\hat{\alpha}})$ test is written

$$\mathrm{Z}(\mathrm{t}_{\hat{\alpha}}) = (\hat{\gamma}_0/\hat{\lambda})^{1/2}\mathrm{t}_{\hat{\alpha}} - \frac{1}{2}T\hat{\sigma}(\hat{\lambda} - \hat{\gamma}_0)/(\mathrm{s}\hat{\lambda}^{1/2})$$

and its limiting distribution is derived as

$$\frac{\frac{1}{2}\{[B(1)]^2 - 1\}}{\{\int_0^1 [B(x)]^2 dx\}^{1/2}}$$

When you test the regression model $y_t = \mu + \alpha y_{t-1} + u_t$ for the true random walk process (single mean case), the limiting distribution of the statistic $Z(\hat{\alpha})$ is written

$$\frac{\frac{1}{2}\{[B(1)]^2 - 1\} - B(1)\int_0^1 B(x)dx}{\int_0^1 [B(x)]^2 dx - \left[\int_0^1 B(x)dx\right]^2}$$

while the limiting distribution of the statistic $\mathrm{Z}(\mathrm{t}_{\hat{\alpha}})$ is given by

$$\frac{\frac{1}{2}\{[B(1)]^2 - 1\} - B(1)\int_0^1 B(x)dx}{\{\int_0^1 [B(x)]^2 dx - \left[\int_0^1 B(x)dx\right]^2\}^{1/2}}$$

Finally, the limiting distribution of the Phillips-Perron test for the random walk with drift process $y_t = \mu + y_{t-1} + u_t$ (trend case) can be derived as

$$[0 \quad c \quad 0]\, V^{-1} \begin{bmatrix} B(1) \\ \frac{B(1)^2 - 1}{2} \\ B(1) - \int_0^1 B(x)dx \end{bmatrix}$$

where $c = 1$ for $Z(\hat{\alpha})$ and $c = \frac{1}{\sqrt{Q}}$ for $\mathrm{Z}(\mathrm{t}_{\hat{\alpha}})$,

$$V = \begin{bmatrix} 1 & \int_0^1 B(x)dx & \frac{1}{2} \\ \int_0^1 B(x)dx & \int_0^1 B(x)^2 dx & \int_0^1 xB(x)dx \\ \frac{1}{2} & \int_0^1 xB(x)dx & \frac{1}{3} \end{bmatrix}$$

$$Q = [0 \quad c \quad 0]\, V^{-1} \begin{bmatrix} 0 \\ c \\ 0 \end{bmatrix}$$

When several variables $\mathbf{z}_t = (z_{1t}, \cdots, z_{kt})'$ are cointegrated, there exists

a (k×1) cointegrating vector **c** such that **c'z**$_t$ is stationary and **c** is a nonzero vector. The residual based cointegration test is based on the following regression model:

$$y_t = \beta_1 + \mathbf{x}_t'\beta + u_t$$

where $y_t = z_{1t}$, $\mathbf{x}_t = (z_{2t}, \cdots, z_{kt})'$, and $\beta = (\beta_2, \cdots, \beta_k)'$. You can estimate the consistent cointegrating vector using OLS if all variables are difference stationary, that is, I(1). The Phillips-Ouliaris test is computed using the OLS residuals from the preceding regression model, and it performs the test for the null hypothesis of no cointegration. The estimated cointegrating vector is $\hat{\mathbf{c}} = (1, -\hat{\beta}_2, \cdots, -\hat{\beta}_k)'$.

Since the AUTOREG procedure does not produce the *p*-value of the cointegration test, you need to refer to the tables by Phillips and Ouliaris (1990). Before you apply the cointegration test, you might perform the unit root test for each variable.

Predicted Values

The AUTOREG procedure can produce two kinds of predicted values for the response series and corresponding residuals and confidence limits. The residuals in both cases are computed as the actual value minus the predicted value. In addition, when GARCH models are estimated, the AUTOREG procedure can output predictions of the conditional error variance.

Predicting the Unconditional Mean

The first type of predicted value is obtained from only the structural part of the model, $\mathbf{x}_t'\mathbf{b}$. These are useful in predicting values of new response time series, which are assumed to be described by the same model as the current response time series. The predicted values, residuals, and upper and lower confidence limits for the structural predictions are requested by specifying the PREDICTEDM=, RESIDUALM=, UCLM=, or LCLM= options in the OUTPUT statement. The ALPHACLM= option controls the confidence level for UCLM= and LCLM=. These confidence limits are for estimation of the mean of the dependent variable, $\mathbf{x}_t'\mathbf{b}$, where $\mathbf{x}_t$ is the column vector of independent variables at observation *t*.

The predicted values are computed as

$$\hat{y}_t = \mathbf{x}_t'\mathbf{b}$$

and the upper and lower confidence limits as

$$\hat{u}_t = \hat{y}_t + t_{\alpha/2}\mathrm{v}$$

$$\hat{l}_t = \hat{y}_t - t_{\alpha/2}\mathrm{v}$$

where v^2 is an estimate of the variance of $\hat{y}_t$ and $t_{\alpha/2}$ is the upper α/2 percentage point of the *t* distribution.

$$\mathrm{Prob}(\mathrm{T} > \mathrm{t}_{\alpha/2}) = \alpha/2$$

where *T* is an observation from a *t* distribution with *q* degrees of freedom. The value of α can be set with the ALPHACLM= option. The degrees of freedom parameter, *q*, is taken to be the number of observations minus the number of free parameters in the regression and autoregression parts of the model. For the YW estimation method, the value of v is calculated as

$$\mathrm{v} = \sqrt{\mathrm{s}^2\mathbf{x}_\mathrm{t}'(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{x}_\mathrm{t}}$$

where s^2 is the error sum of squares divided by *q*. For the ULS and ML methods, it is calculated as

$$\mathrm{v} = \sqrt{\mathrm{s}^2\mathbf{x}_\mathrm{t}'\mathbf{W}\mathbf{x}_\mathrm{t}}$$

where **W** is the $k \times k$ submatrix of $(\mathbf{J}'\mathbf{J})^{-1}$ that corresponds to the regression parameters. For details, see “Computational Methods” earlier in this chapter.

Predicting Future Series Realizations

The other predicted values use both the structural part of the model and the predicted values of the error process. These conditional mean values are useful in predicting future values of the current response time series. The predicted values, residuals, and upper and lower confidence limits for future observations conditional on past values are requested by the PREDICTED=, RESIDUAL=, UCL=, or LCL= options in the OUTPUT statement. The ALPHACLI= option controls the confidence level for UCL= and LCL=. These confidence limits are for the predicted value,

$$\tilde{y}_t = \mathbf{x}_t'\mathbf{b} + \nu_{t|t-1}$$

where $\mathbf{x}_t$ is the vector of independent variables and $\nu_{t|t-1}$ is the minimum variance linear predictor of the error term given the available past values of $\nu_{t-j}, j = 1, 2, \ldots, t-1$, and the autoregressive

model for ν_t. If the *m* previous values of the structural residuals are available, then

$$\nu_{t|t-1} = -\hat{\varphi}_1\nu_{t-1} - \ldots - \hat{\varphi}_m\nu_{t-m}$$

where $\hat{\varphi}_1, \ldots, \hat{\varphi}_m$ are the estimated AR parameters. The upper and lower confidence limits are computed as

$$\tilde{u}_t = \tilde{y}_t + t_{\alpha/2}\mathrm{v}$$

$$\tilde{l}_t = \tilde{y}_t - t_{\alpha/2}\mathrm{v}$$

where v, in this case, is computed as

$$\mathrm{v} = \sqrt{\mathrm{s}^2(\mathbf{x}_\mathrm{t}'(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{x}_\mathrm{t} + \mathrm{r})}$$

where the value rs^2 is the estimate of the variance of $\nu_{t|t-1}$. At the start of the series, and after missing values, *r* is generally greater than 1. See “Predicting the Conditional Variance” for computational details of *r*. The plot of residuals and confidence limits in Example 12.4 later in this chapter illustrates this behavior.

Except to adjust the degrees of freedom for the error sum of squares, the preceding formulas do not account for the fact that the autoregressive parameters are estimated. In particular, the confidence limits are likely to be somewhat too narrow. In large samples, this is probably not an important effect, but it may be appreciable in small samples. Refer to Harvey (1981) for some discussion of this problem for AR(1) models.

Note that at the beginning of the series (the first *m* observations, where *m* is the value of the NLAG= option) and after missing values, these residuals do not match the residuals obtained by using OLS on the transformed variables. This is because, in these cases, the predicted noise values must be based on less than a complete set of past noise values and, thus, have larger variance. The GLS transformation for these observations includes a scale factor as well as a linear combination of past values. Put another way, the $\mathbf{L}^{-1}$ matrix defined in the section "Computational Methods" has the value 1 along the diagonal, except for the first *m* observations and after missing values.

Predicting the Conditional Variance

The GARCH process can be written

$$\epsilon_t^2 = \omega + \sum_{i=1}^{n} (\alpha_i + \gamma_i)\epsilon_{t-i}^2 - \sum_{j=1}^{p} \gamma_j \eta_{t-j} + \eta_t$$

where $\eta_t = \epsilon_t^2 - h_t$and $n = \max(p, q)$. This representation shows that the squared residual ϵ_t^2 follows an ARMA(*n*,*p*) process. Then for any $d > 0$, the conditional expectations are as follows:

$$\mathbf{E}(\epsilon_{t+d}^2|\Psi_t) = \omega + \sum_{i=1}^{n} (\alpha_i + \gamma_i)\mathbf{E}(\epsilon_{t+d-i}^2|\Psi_t) - \sum_{j=1}^{p} \gamma_j \mathbf{E}(\eta_{t+d-j}|\Psi_t)$$

The *d*-step-ahead prediction error, ξ_{t+d} = $y_{t+d} - y_{t+d|t}$, has the conditional variance

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sum_{j=0}^{d-1} g_j^2 \sigma_{t+d-j|t}^2$$

where

$$\sigma_{t+d-j|t}^2 = \mathbf{E}(\epsilon_{t+d-j}^2|\Psi_t)$$

Coefficients in the conditional *d*-step prediction error variance are calculated recursively using the following formula:

$$g_j = -\varphi_1 g_{j-1} - \ldots - \varphi_m g_{j-m}$$

where $g_0 = 1$ and $g_j = 0$ if $j < 0$; φ_1, …, φ_m are autoregressive parameters. Since the parameters are not known, the conditional variance is computed using the estimated autoregressive parameters. The *d*-step-ahead prediction error variance is simplified when there are no autoregressive terms:

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sigma_{t+d|t}^2$$

Therefore, the one-step-ahead prediction error variance is equivalent to the conditional error variance defined in the GARCH process:

$$h_t = E(\epsilon_t^2|\Psi_{t-1}) = \sigma_{t|t-1}^2$$

Note that the conditional prediction error variance of the EGARCH and GARCH-M models cannot be calculated using the preceding formula. Therefore, the confidence intervals for the predicted values are computed assuming the homoscedastic conditional error variance. That is, the conditional prediction error variance is identical to the unconditional prediction error variance:

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \mathbf{V}(\xi_{t+d}) = \sigma^2 \sum_{j=0}^{d-1} g_j^2$$

since $\sigma_{t+d-j|t}^2 = \sigma^2$. You can compute $s^2 r$, which is the second term of the variance for the predicted value $\tilde{y}_t$ explained previously in "Predicting Future Series Realizations," using the formula $\sigma^2 \sum_{j=0}^{d-1} g_j^2$; r is estimated from $\sum_{j=0}^{d-1} g_j^2$ using the estimated autoregressive parameters.

Consider the following conditional prediction error variance:

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sigma^2 \sum_{j=0}^{d-1} g_j^2 + \sum_{j=0}^{d-1} g_j^2 (\sigma_{t+d-j|t}^2 - \sigma^2)$$

The second term in the preceding equation can be interpreted as the noise from using the homoscedastic conditional variance when the errors follow the GARCH process. However, it is expected that if the GARCH process is covariance stationary, the difference between the conditional prediction error variance and the unconditional prediction error variance disappears as the forecast horizon *d* increases.

OUT= Data Set

The output SAS data set produced by the OUTPUT statement contains all the variables in the input data set and the new variables specified by the OUTPUT statement options. See the section "OUTPUT Statement" earlier in this chapter for information on the output variables that can be created. The output data set contains one observation for each observation in the input data set.

OUTEST= Data Set

The OUTEST= data set contains all the variables used in any MODEL statement. Each regressor variable contains the estimate for the corresponding regression parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

_A_*i*	the *i*th order autoregressive parameter estimate. There are *m* such variables _A_1 through _A_*m*, where *m* is the value of the NLAG= option.
_AH_*i*	the *i*th order ARCH parameter estimate, if the GARCH= option is specified. There are *q* such variables _AH_1 through _AH_*q*, where *q* is the value of the Q= option. The variable _AH_0 contains the estimate of ω.
DELTA	the estimated mean parameter for the GARCH-M model, if a GARCH-in-mean model is specified
DEPVAR	the name of the dependent variable
_GH_*i*	the *i*th order GARCH parameter estimate, if the GARCH= option is specified. There are *p* such variables _GH_1 through _GH_*p*, where *p* is the value of the P= option.
INTERCEPT	the intercept estimate. INTERCEP contains a missing value for models for which the NOINT option is specified.
METHOD	the estimation method that is specified in the METHOD= option
MODEL	the label of the MODEL statement if one is given, or blank otherwise
MSE	the value of the mean square error for the model
NAME	the name of the row of covariance matrix for the parameter estimate, if the COVOUT option is specified
LIKLHD	the log likelihood value of the GARCH model
SSE	the value of the error sum of squares
STDERR	standard error of the parameter estimate, if the COVOUT option is specified
THETA	the estimate of the θ parameter in the EGARCH model, if an EGARCH model is specified
TYPE	OLS for observations containing parameter estimates, or COV for observations containing covariance matrix elements.

The OUTEST= data set contains one observation for each MODEL statement giving the parameter estimates for that model. If the COVOUT option is specified, the OUTEST= data set includes additional observations for each MODEL statement giving the rows of the covariance of parameter estimates matrix. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter associated with that row of the covariance matrix.

Printed Output

The AUTOREG procedure prints the following items:

1. the name of the dependent variable
2. the ordinary least-squares estimates
3. estimates of autocorrelations, which include the estimates of the autocovariances, the autocorrelations, and (if there is sufficient space) a graph of the autocorrelation at each LAG
4. if the PARTIAL option is specified, the partial autocorrelations
5. the preliminary MSE, which results from solving the Yule-Walker equations. This is an estimate of the final MSE.
6. the estimates of the autoregressive parameters (Coefficient), their standard errors (Std Error), and the ratio of estimate to standard error (*t* Ratio).
7. the statistics of fit are printed for the final model. These include the error sum of squares (SSE), the degrees of freedom for error (DFE), the mean square error (MSE), the root mean square error (Root MSE), the Schwarz information criterion (SBC), the Akaike information criterion (AIC), the regression R^2 (Reg Rsq), and the total R^2 (Total Rsq). For GARCH models, the following additional items are printed:

   - the value of the log likelihood function
   - the number of observations that are used in estimation (OBS)
   - the unconditional variance (UVAR)
   - the normality test statistic and its *p*-value

8. the parameter estimates for the structural model (B Value), a standard error estimate (Std Error), the ratio of estimate to standard error (*t* Ratio), and an approximation to the significance probability for the parameter being 0 (Approx Prob)
9. the regression parameter estimates, printed again assuming that the autoregressive parameter estimates are known to be correct. The Std Error and related statistics for the regression estimates will, in general, be different when the autoregressive parameters are assumed to be given.

ODS Table Names

PROC AUTOREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, “Using the Output Delivery System.”

Table 12.1. ODS Tables Produced in PROC AUTOREG

ODS Table Name	Description	Option
ODS Tables Created by the Model Statement		
FitSummary	Summary of regression	default
SummaryDepVarCen	Summary of regression (centered dependent var)	CENTER
SummaryNoIntercept	Summary of regression (no intercept)	NOINT
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW
PreMSE	Preliminary MSE	NLAG=
Dependent	Dependent variable	default
DependenceEquations	Linear dependence equation	
ARCHTest	Q and LM Tests for ARCH Disturbances	ARCHTEST
ChowTest	Chow Test and Predictive Chow Test	CHOW= PCHOW=
Godfrey	Godfrey's Serial Correlation Test	GODFREY GODFREY=
PhilPerron	Phillips-Perron Unit Root Test	STATIONARITY= (PHILIPS<=()>) (no regressor)
PhilOul	Phillips-Ouliaris Cointegration Test	STATIONARITY= (PHILIPS<=()>) (has regressor)
ResetTest	Ramsey's RESET Test	RESET
ARParameterEstimates	Estimates of Autoregressive Parameters	NLAG=
CorrGraph	Estimates of Autocorrelations	NLAG=
BackStep	Backward Elimination of Autoregressive Terms	BACKSTEP
ExpAutocorr	Expected Autocorrelations	NLAG=
IterHistory	Iteration History	ITPRINT
ParameterEstimates	Parameter Estimates	default
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=
PartialAutoCorr	Partial autocorrelation	PARTIAL
CovB	Covariance of Parameter Estimates	COVB
CorrB	Correlation of Parameter Estimates	CORRB
CholeskyFactor	Cholesky Root of Gamma	ALL
Coefficients	Coefficients for First NLAG Observations	COEF
GammaInverse	Gamma Inverse	GINV
ConvergenceStatus	Convergence Status table	default
DWTestProb	Durbin-Watson Statistics	DW=

Table 12.1. (continued)

ODS Table Name	Description	Option
	ODS Tables Created by the Restrict Statement	
Restrict	Restriction table	default
	ODS Tables Created by the Test Statement	
FTest	F test	default
WaldTest	Wald test	TYPE=WALD

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the AUTOREG procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS." By default, only the residual, predicted vs actual, and autocorrelation of residuals plots are produced. If, in addition to the ODS GRAPHICS statement, you also specify the ALL option in either the PROC AUTOREG statement or MODEL statement, all plots are created. However, if the NLAG= option is specified, the Cook's D plot and the studentized residuals plot are not produced.

ODS Graph Names

PROC AUTOREG assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 12.2.

Table 12.2. ODS Graphics Produced by PROC AUTOREG

ODS Graph Name	Plot Description	Option
ACFPlot	Autocorrelation of residuals	Default
ActualByPredicted	Predicted vs actual plot	Default
CooksD	Cook's D plot	ALL (no NLAG=)
IACFPlot	Inverse autocorrelation of residuals	ALL
QQPlot	QQ plot of residuals	ALL
PACFPlot	Partial autocorrelation of residuals	ALL
ResidualHistogram	Histogram of the residuals	ALL
ResidualPlot	Residual plot	Default
StudentResidualPlot	Studentized Residual plot	ALL (no NLAG=)
WhiteNoisePlot	Tests for White Noise Residuals	ALL

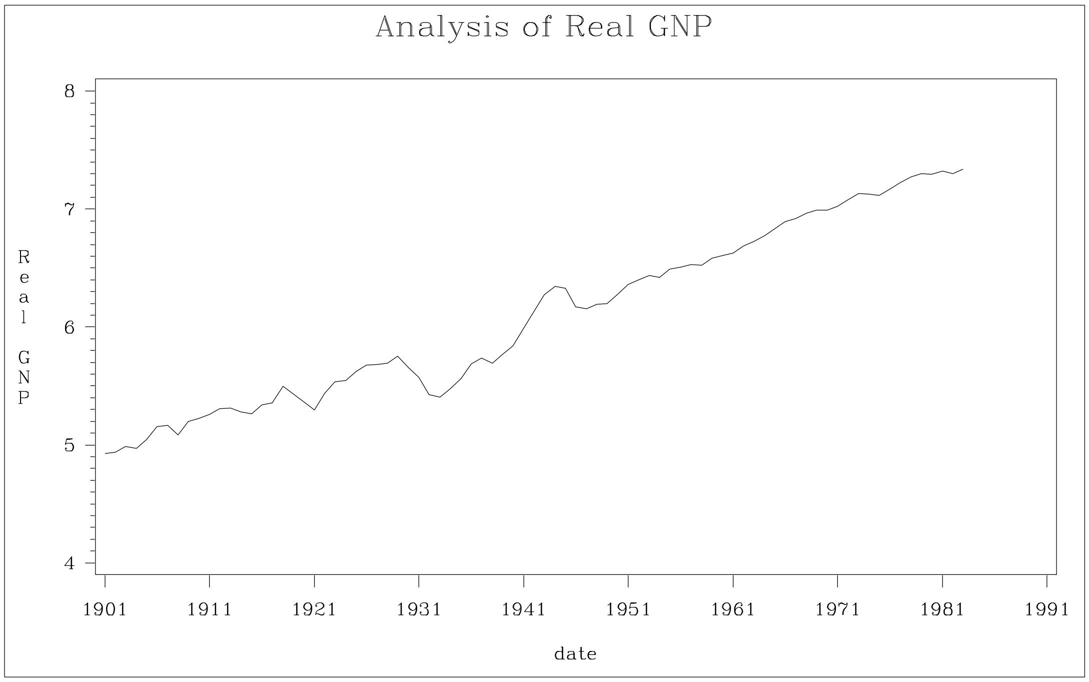
Examples

Example 12.1. Analysis of Real Output Series

In this example, the annual real output series is analyzed over the period 1901 to 1983 (Gordon 1986, pp 781-783). With the DATA step, the original data is transformed using the natural logarithm, and the differenced series DY is created for further analysis. The log of real output is plotted in Output 12.1.1.

```
title 'Analysis of Real GNP';
data gnp;
   date = intnx( 'year', '01jan1901'd, _n_-1 );
   format date year4.;
   input x @@;
   y  = log(x);
   dy = dif(y);
   t  = _n_;
   label y  = 'Real GNP'
         dy = 'First Difference of Y'
         t  = 'Time Trend';
datalines;
... datalines omitted ...
;

proc gplot data=gnp;
   plot y * date /
       haxis='01jan1901'd '01jan1911'd '01jan1921'd '01jan1931'd
             '01jan1941'd '01jan1951'd '01jan1961'd '01jan1971'd
             '01jan1981'd '01jan1991'd;
   symbol i=join;
run;
```

Output 12.1.1. Real Output Series: 1901 - 1983



The (linear) trend-stationary process is estimated using the following form:

$$y_t = \beta_0 + \beta_1 t + \nu_t$$

where

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1} - \varphi_2 \nu_{t-2}$$

$$\epsilon_t \sim \mathrm{IN}(0, \sigma_\epsilon)$$

The preceding trend-stationary model assumes that uncertainty over future horizons is bounded since the error term, ν_t, has a finite variance. The maximum likelihood AR estimates are shown in Output 12.1.2.

```
proc autoreg data=gnp;
   model y = t / nlag=2 method=ml;
run;
```

Output 12.1.2. Estimating the Linear Trend Model

```
                              The AUTOREG Procedure

                           Maximum Likelihood Estimates

          SSE                 0.23954331    DFE                      79
          MSE                    0.00303    Root MSE            0.05507
          SBC                 -230.39355    AIC              -240.06891
          Regress R-Square        0.8645    Total R-Square       0.9947
          Durbin-Watson           1.9935


                                   Standard              Approx   Variable
  Variable       DF    Estimate       Error   t Value   Pr > |t|   Label

  Intercept       1      4.8206      0.0661     72.88     <.0001
  t               1      0.0302    0.001346     22.45     <.0001   Time Trend
  AR1             1     -1.2041      0.1040    -11.58     <.0001
  AR2             1      0.3748      0.1039      3.61     0.0005


                    Autoregressive parameters assumed given.

                                   Standard              Approx   Variable
  Variable       DF    Estimate       Error   t Value   Pr > |t|   Label

  Intercept       1      4.8206      0.0661     72.88     <.0001
  t               1      0.0302    0.001346     22.45     <.0001   Time Trend
```

Nelson and Plosser (1982) failed to reject the hypothesis that macroeconomic time series are nonstationary and have no tendency to return to a trend line. In this context, the simple random walk process can be used as an alternative process:

$$y_t = \beta_0 + y_{t-1} + \nu_t$$

where $\nu_t = \epsilon_t$ and $y_0 = 0$. In general, the difference-stationary process is written as

$$\phi(L)(1 - L)y_t = \beta_0\phi(1) + \theta(L)\epsilon_t$$

where L is the lag operator. You can observe that the class of a difference-stationary process should have at least one unit root in the AR polynomial $\phi(L)(1 - L)$.

The Dickey-Fuller procedure is used to test the null hypothesis that the series has a unit root in the AR polynomial. Consider the following equation for the augmented Dickey-Fuller test:

$$\Delta y_t = \beta_0 + \delta t + \beta_1 y_{t-1} + \sum_{i=1}^{m} \gamma_i \Delta y_{t-i} + \epsilon_t$$

where $\Delta = 1 - L$. The test statistic τ_τ is the usual *t* ratio for the parameter estimate $\hat{\beta}_1$, but the τ_τ does not follow a *t* distribution.

The %DFTEST macro computes the test statistic τ_τ and its *p* value to perform the Dickey-Fuller test. The default value of *m* is 3, but you can specify *m* with the AR= option. The option TREND=2 implies that the Dickey-Fuller test equation contains linear time trend. See Chapter 4, “SAS Macros and Functions,” for details.

```
%dftest(gnp,y,trend=2,outstat=stat1)

proc print data=stat1;
run;
```

The augmented Dickey-Fuller test indicates that the output series may have a difference-stationary process. The statistic _TAU_ has a value of -2.61903 and its *p*-value is 0.29104. See Output 12.1.3.

Output 12.1.3. Augmented Dickey-Fuller Test Results

```
                                                      I
             _        _                               n
             S        D                               t
    _        T        E    _                          e
    T        A        P    N                   _      r
    Y        T        V    A                   M      c      A            t
O   P        U        A    M                   S      e      R            i
b   E        S        R    E                   E      p      _            m
s   _        _        _    _                   _      t      V            e

1  OLS  0 Converged  AR_V             .003198469   0.76919  -1  0.004816233
2  COV  0 Converged  AR_V  Intercept  .003198469   0.08085   .  0.000513286
3  COV  0 Converged  AR_V  time       .003198469   0.00051   .  0.000003387
4  COV  0 Converged  AR_V  DLAG_V     .003198469  -0.01695   .  -.000108543
5  COV  0 Converged  AR_V  AR_V1      .003198469   0.00549   .  0.000035988
6  COV  0 Converged  AR_V  AR_V2      .003198469   0.00842   .  0.000054197
7  COV  0 Converged  AR_V  AR_V3      .003198469   0.01056   .  0.000067710


                                                                    _
                                                           _        P
       D                                      _            T  _     V
       L         A         A          A       N      _     R  D     A
       A         R         R          R       O      T     E  L     L
O      G         _         _          _       B      A     N  A     U
b      _         V         V          V       S      U     D  G     E
s      V         1         2          3       _      _     _  _     _

1  -0.15629   0.37194   0.025483  -0.082422  79  -2.61903  2  1  0.27321
2  -0.01695   0.00549   0.008422   0.010556  79  -2.61903  2  1  0.27321
3  -0.00011   0.00004   0.000054   0.000068  79  -2.61903  2  1  0.27321
4   0.00356  -0.00120  -0.001798  -0.002265  79  -2.61903  2  1  0.27321
5  -0.00120   0.01242  -0.003455   0.002095  79  -2.61903  2  1  0.27321
6  -0.00180  -0.00346   0.014238  -0.002910  79  -2.61903  2  1  0.27321
7  -0.00226   0.00209  -0.002910   0.013538  79  -2.61903  2  1  0.27321
```

The AR(1) model for the differenced series DY is estimated using the maximum likelihood method for the period 1902 to 1983. The difference-stationary process is written

$$\Delta y_t = \beta_0 + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1}$$

The estimated value of φ_1 is -0.297 and that of β_0 is 0.0293. All estimated values are statistically significant.

```
proc autoreg data=gnp;
   model dy = / nlag=1 method=ml;
run;
```

Output 12.1.4. Estimating the Differenced Series with AR(1) Error

```
                           The AUTOREG Procedure

                       Maximum Likelihood Estimates

    SSE                   0.27107673    DFE                      80
    MSE                      0.00339    Root MSE            0.05821
    SBC                   -226.77848    AIC              -231.59192
    Regress R-Square          0.0000    Total R-Square       0.0900
    Durbin-Watson             1.9268


                                      Standard                 Approx
 Variable        DF      Estimate        Error    t Value    Pr > |t|

 Intercept        1        0.0293     0.009093       3.22      0.0018
 AR1              1       -0.2967       0.1067      -2.78      0.0067


                  Autoregressive parameters assumed given.

                                      Standard                 Approx
 Variable        DF      Estimate        Error    t Value    Pr > |t|

 Intercept        1        0.0293     0.009093       3.22      0.0018
```

Example 12.2. Comparing Estimates and Models

In this example, the Grunfeld series are estimated using different estimation methods. Refer to Maddala (1977) for details of the Grunfeld investment data set. For comparison, the Yule-Walker method, the ULS method, and maximum likelihood method estimates are shown. With the DWPROB option, the *p*-value of the Durbin-Watson statistic is printed. The Durbin-Watson test indicates the positive autocorrelation of the regression residuals.

```
title 'Grunfeld''s Investment Models Fit with Autoregressive Errors';
data grunfeld;
   input year gei gef gec;
   label gei = 'Gross investment GE'
         gec = 'Lagged Capital Stock GE'
         gef = 'Lagged Value of GE shares';
datalines;
  ... data lines omitted ...
;

proc autoreg data=grunfeld;
   model gei = gef gec / nlag=1 dwprob;
   model gei = gef gec / nlag=1 method=uls;
   model gei = gef gec / nlag=1 method=ml;
run;
```

The printed output produced by each of the MODEL statements is shown in Output 12.2.1 through Output 12.2.4.

Output 12.2.1. OLS Analysis of Residuals

```
           Grunfeld's Investment Models Fit with Autoregressive Errors

                              The AUTOREG Procedure

                    Dependent Variable                    gei
                                          Gross investment GE


                         Ordinary Least Squares Estimates

          SSE                  13216.5878    DFE                        17
          MSE                   777.44634    Root MSE             27.88272
          SBC                  195.614652    AIC                192.627455
          Regress R-Square         0.7053    Total R-Square         0.7053
          Durbin-Watson            1.0721    Pr < DW                0.0038
          Pr > DW                  0.9962


                             Standard           Approx
Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

Intercept     1   -9.9563   31.3742   -0.32   0.7548
gef           1    0.0266    0.0156    1.71   0.1063 Lagged Value of GE shares
gec           1    0.1517    0.0257    5.90   <.0001 Lagged Capital Stock GE


                          Estimates of Autocorrelations

 Lag   Covariance    Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

   0        660.8       1.000000   |                    |********************|
   1        304.6       0.460867   |                    |*********           |


                           Preliminary MSE       520.5
```

Output 12.2.2. Regression Results Using Default Yule-Walker Method

```
           Grunfeld's Investment Models Fit with Autoregressive Errors

                              The AUTOREG Procedure

                      Estimates of Autoregressive Parameters

                                            Standard
                   Lag     Coefficient         Error    t Value

                     1       -0.460867      0.221867      -2.08


                              Yule-Walker Estimates

          SSE                  10238.2951    DFE                        16
          MSE                   639.89344    Root MSE             25.29612
          SBC                  193.742396    AIC                189.759467
          Regress R-Square         0.5717    Total R-Square         0.7717
          Durbin-Watson            1.3321    Pr < DW                0.0232
          Pr > DW                  0.9768


                             Standard           Approx
Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

Intercept     1  -18.2318   33.2511   -0.55   0.5911
gef           1    0.0332    0.0158    2.10   0.0523 Lagged Value of GE shares
gec           1    0.1392    0.0383    3.63   0.0022 Lagged Capital Stock GE
```

Output 12.2.3. Regression Results Using Unconditional Least Squares Method

```
          Grunfeld's Investment Models Fit with Autoregressive Errors

                            The AUTOREG Procedure

                    Estimates of Autoregressive Parameters

                                            Standard
                 Lag     Coefficient           Error    t Value

                   1       -0.460867        0.221867      -2.08


        Algorithm converged.


                    Unconditional Least Squares Estimates

         SSE                  10220.8455    DFE                      16
         MSE                   638.80284    Root MSE           25.27455
         SBC                  193.756692    AIC              189.773763
         Regress R-Square         0.5511    Total R-Square       0.7721
         Durbin-Watson            1.3523


                                Standard           Approx
 Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

 Intercept     1  -18.6582   34.8101   -0.54   0.5993
 gef           1    0.0339    0.0179    1.89   0.0769 Lagged Value of GE shares
 gec           1    0.1369    0.0449    3.05   0.0076 Lagged Capital Stock GE
 AR1           1   -0.4996    0.2592   -1.93   0.0718


                   Autoregressive parameters assumed given.

                                Standard           Approx
 Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

 Intercept     1  -18.6582   33.7567   -0.55   0.5881
 gef           1    0.0339    0.0159    2.13   0.0486 Lagged Value of GE shares
 gec           1    0.1369    0.0404    3.39   0.0037 Lagged Capital Stock GE
```

Output 12.2.4. Regression Results Using Maximum Likelihood Method

```
           Grunfeld's Investment Models Fit with Autoregressive Errors

                                The AUTOREG Procedure

                        Estimates of Autoregressive Parameters

                                                Standard
                     Lag      Coefficient          Error     t Value

                       1        -0.460867       0.221867       -2.08


         Algorithm converged.


                             Maximum Likelihood Estimates

          SSE                   10229.2303    DFE                       16
          MSE                    639.32689    Root MSE            25.28491
          SBC                   193.738877    AIC               189.755947
          Regress R-Square          0.5656    Total R-Square        0.7719
          Durbin-Watson             1.3385


                                 Standard            Approx
  Variable     DF   Estimate       Error  t Value  Pr > |t|  Variable Label

  Intercept     1   -18.3751     34.5941    -0.53    0.6026
  gef           1     0.0334      0.0179     1.87    0.0799  Lagged Value of GE shares
  gec           1     0.1385      0.0428     3.23    0.0052  Lagged Capital Stock GE
  AR1           1    -0.4728      0.2582    -1.83    0.0858


                       Autoregressive parameters assumed given.

                                 Standard            Approx
  Variable     DF   Estimate       Error  t Value  Pr > |t|  Variable Label

  Intercept     1   -18.3751     33.3931    -0.55    0.5897
  gef           1     0.0334      0.0158     2.11    0.0512  Lagged Value of GE shares
  gec           1     0.1385      0.0389     3.56    0.0026  Lagged Capital Stock GE
```

Example 12.3. Lack of Fit Study

Many time series exhibit high positive autocorrelation, having the smooth appearance of a random walk. This behavior can be explained by the partial adjustment and adaptive expectation hypotheses.

Short-term forecasting applications often use autoregressive models because these models absorb the behavior of this kind of data. In the case of a first-order AR process where the autoregressive parameter is exactly 1 (a *random walk*), the best prediction of the future is the immediate past.

PROC AUTOREG can often greatly improve the fit of models, not only by adding additional parameters but also by capturing the random walk tendencies. Thus, PROC AUTOREG can be expected to provide good short-term forecast predictions.

However, good forecasts do not necessarily mean that your structural model contributes anything worthwhile to the fit. In the following example, random noise is fit to part of a sine wave. Notice that the structural model does not fit at all, but the autoregressive process does quite well and is very nearly a first difference (A(1) = -.976).

```
title1 'Lack of Fit Study';
title2 'Fitting White Noise Plus Autoregressive Errors to a Sine Wave';

data a;
   pi=3.14159;
   do time = 1 to 75;
      if time > 75 then y = .;
      else y = sin( pi * ( time / 50 ) );
      x = ranuni( 1234567 );
      output;
      end;
run;

proc autoreg data=a;
   model y = x / nlag=1;
   output out=b p=pred pm=xbeta;
run;

proc gplot data=b;
   plot y*time=1 pred*time=2 xbeta*time=3 / overlay;
   symbol1  v='none' i=spline;
   symbol2  v=triangle;
   symbol3  v=circle;
run;
```

The printed output produced by PROC AUTOREG is shown in Output 12.3.1 and Output 12.3.2. Plots of observed and predicted values are shown in Output 12.3.3.

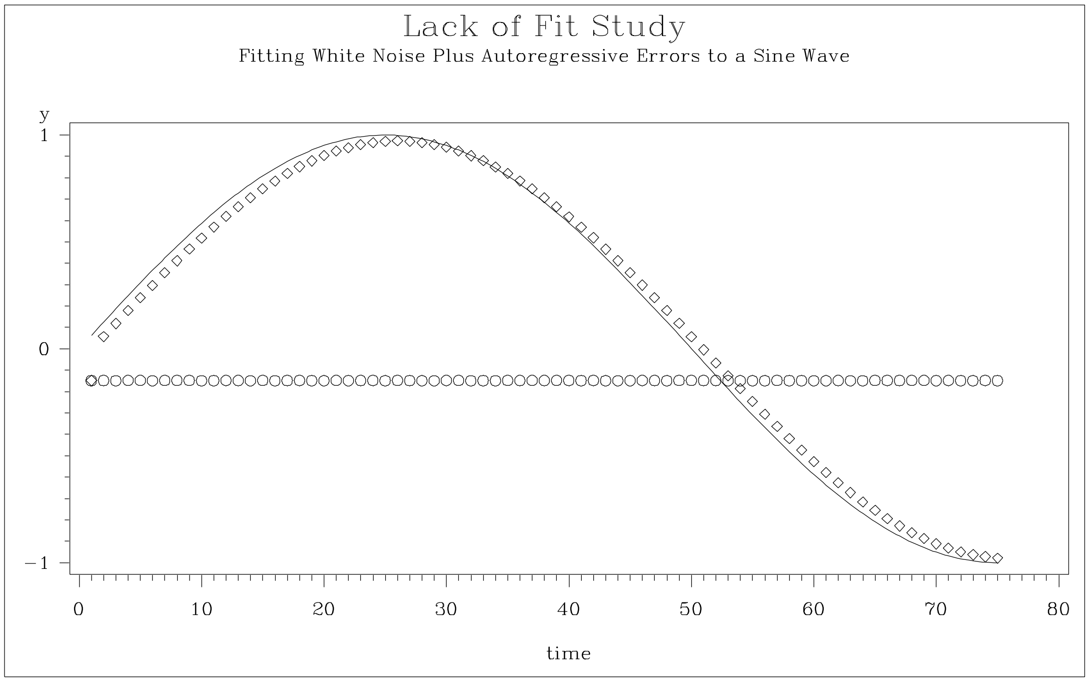
Output 12.3.1. Results of OLS Analysis: No Autoregressive Model Fit

```
                               Lack of Fit Study
          Fitting White Noise Plus Autoregressive Errors to a Sine Wave

                             The AUTOREG Procedure

                            Dependent Variable    y


                        Ordinary Least Squares Estimates

          SSE                34.8061005    DFE                       73
          MSE                   0.47680    Root MSE             0.69050
          SBC                163.898598    AIC               159.263622
          Regress R-Square       0.0008    Total R-Square        0.0008
          Durbin-Watson          0.0057


                                         Standard                 Approx
       Variable        DF     Estimate      Error    t Value    Pr > |t|

       Intercept        1       0.2383     0.1584       1.50      0.1367
       x                1      -0.0665     0.2771      -0.24      0.8109


                         Estimates of Autocorrelations

 Lag   Covariance    Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

   0       0.4641       1.000000   |                    |********************|
   1       0.4531       0.976386   |                    |********************|


                          Preliminary MSE      0.0217
```

Output 12.3.2. Regression Results with AR(1) Error Correction

```
                           Lack of Fit Study
         Fitting White Noise Plus Autoregressive Errors to a Sine Wave

                        The AUTOREG Procedure

                 Estimates of Autoregressive Parameters

                                         Standard
                 Lag      Coefficient       Error    t Value

                   1        -0.976386    0.025460     -38.35


                        Yule-Walker Estimates

         SSE                 0.18304264    DFE                      72
         MSE                    0.00254    Root MSE            0.05042
         SBC                 -222.30643    AIC               -229.2589
         Regress R-Square        0.0001    Total R-Square       0.9947
         Durbin-Watson           0.0942


                                      Standard                 Approx
      Variable        DF     Estimate      Error    t Value    Pr > |t|

      Intercept        1      -0.1473     0.1702      -0.87      0.3898
      x                1    -0.001219     0.0141      -0.09      0.9315
```

Output 12.3.3. Plot of Autoregressive Prediction



Example 12.4. Missing Values

In this example, a pure autoregressive error model with no regressors is used to generate 50 values of a time series. Approximately fifteen percent of the values are randomly chosen and set to missing. The following statements generate the data.

```
title  'Simulated Time Series with Roots:';
title2 ' (X-1.25)(X**4-1.25)';
title3 'With 15% Missing Values';
data ar;
   do i=1 to 550;
      e = rannor(12345);
      n = sum( e, .8*n1, .8*n4, -.64*n5 ); /* ar process  */
      y = n;
      if ranuni(12345) > .85 then y = .;    /* 15% missing */
      n5=n4; n4=n3; n3=n2; n2=n1; n1=n;    /* set lags    */
      if i>500 then output;
      end;
run;
```

The model is estimated using maximum likelihood, and the residuals are plotted with 99% confidence limits. The PARTIAL option prints the partial autocorrelations. The following statements fit the model:

```
proc autoreg data=ar partial;
   model y = / nlag=(1 4 5) method=ml;
   output out=a predicted=p residual=r ucl=u lcl=l alphacli=.01;
run;
```

The printed output produced by the AUTOREG procedure is shown in Output 12.4.1.

Output 12.4.1. Autocorrelation-Corrected Regression Results

```
                     Simulated Time Series with Roots:
                             (X-1.25)(X**4-1.25)
                          With 15% Missing Values

                           The AUTOREG Procedure

                          Dependent Variable    y


                      Ordinary Least Squares Estimates

         SSE                  182.972379    DFE                      40
         MSE                     4.57431    Root MSE            2.13876
         SBC                   181.39282    AIC              179.679248
         Regress R-Square         0.0000    Total R-Square       0.0000
         Durbin-Watson            1.3962


                                         Standard                 Approx
      Variable        DF     Estimate       Error    t Value    Pr > |t|

      Intercept        1      -2.2387      0.3340      -6.70      <.0001


                       Estimates of Autocorrelations

   Lag   Covariance    Correlation   -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1

     0       4.4627       1.000000   |                    |********************|
     1       1.4241       0.319109   |                    |******              |
     2       1.6505       0.369829   |                    |*******             |
     3       0.6808       0.152551   |                    |***                 |
     4       2.9167       0.653556   |                    |*************       |
     5      -0.3816      -0.085519   |                  **|                    |


                                  Partial
                              Autocorrelations

                              1        0.319109
                              4        0.619288
                              5       -0.821179
                          The AUTOREG Procedure

                       Preliminary MSE      0.7609


                  Estimates of Autoregressive Parameters

                                           Standard
               Lag      Coefficient           Error    t Value

                 1        -0.733182        0.089966      -8.15
                 4        -0.803754        0.071849     -11.19
                 5         0.821179        0.093818       8.75


                              Expected
                          Autocorrelations

                          Lag     Autocorr

                            0       1.0000
                            1       0.4204
                            2       0.2480
                            3       0.3160
                            4       0.6903
                            5       0.0228


     Algorithm converged.


                       Maximum Likelihood Estimates

      SSE                 48.4396756    DFE                     37
      MSE                    1.30918    Root MSE           1.14419
      SBC                 146.879013    AIC             140.024725
      Regress R-Square        0.0000    Total R-Square      0.7353
      Durbin-Watson           2.9457


                                        Standard                 Approx
    Variable        DF     Estimate        Error    t Value    Pr > |t|

    Intercept        1      -2.2370       0.5239      -4.27      0.0001
    AR1              1      -0.6201       0.1129      -5.49      <.0001
    AR4              1      -0.7237       0.0914      -7.92      <.0001
    AR5              1       0.6550       0.1202       5.45      <.0001
```

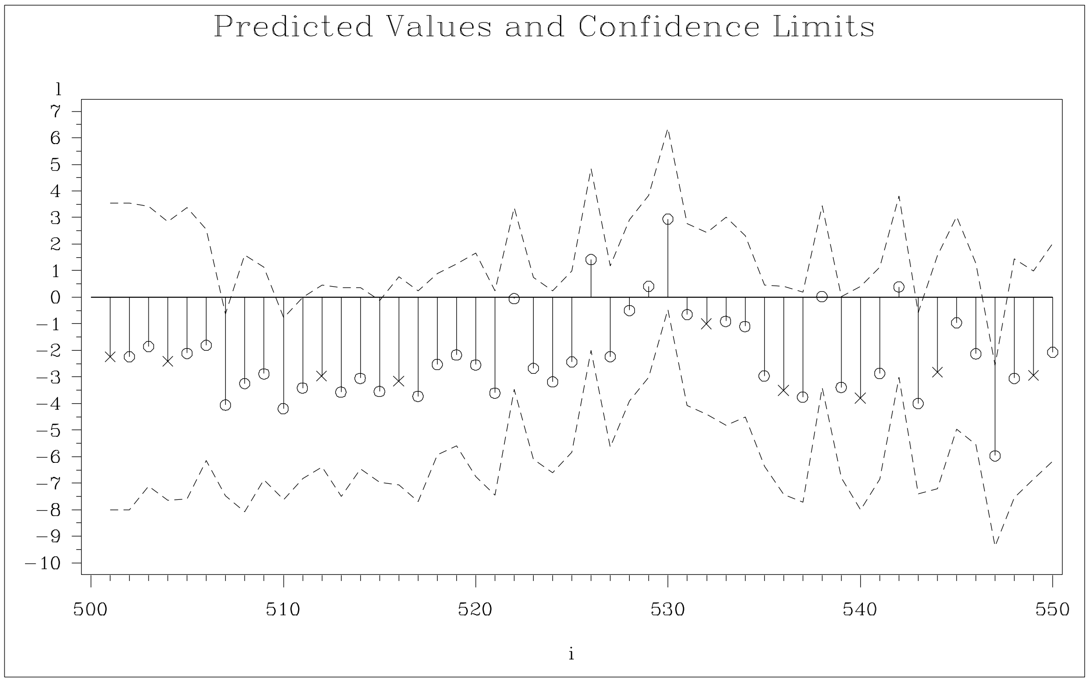
```
                           The AUTOREG Procedure

                                 Expected
                             Autocorrelations

                             Lag     Autocorr

                               0       1.0000
                               1       0.4204
                               2       0.2423
                               3       0.2958
                               4       0.6318
                               5       0.0411


                  Autoregressive parameters assumed given.

                                           Standard                 Approx
      Variable        DF      Estimate        Error     t Value     Pr > |t|

      Intercept        1       -2.2370       0.5225       -4.28       0.0001
```

The following statements plot the residuals and confidence limits:

```
data reshape1;
   set a;
   miss = .;
   if r=. then do;
      miss = p;
      p = .;
      end;
run;

title 'Predicted Values and Confidence Limits';
proc gplot data=reshape1;
   plot l*i=1 miss*i=2 p*i=3 u*i=4 / overlay;
   symbol1  i=join v=none l=2;
   symbol2  i=needle v='X';
   symbol3  i=needle v=circle;
   symbol4  i=join v=none l=2;
run;
```

The plot of the predicted values and the upper and lower confidence limits is shown in Output 12.4.2. Note that the confidence interval is wider at the beginning of the series (when there are no past noise values to use in the forecast equation) and after missing values where, again, there is an incomplete set of past residuals.

Output 12.4.2. Plot of Residuals and Confidence Interval



Example 12.5. Money Demand Model

The following example estimates the log-log money demand equation using the maximum likelihood method. The money demand model contains four explanatory variables. The lagged nominal money stock M1 is divided by the current price level GDF to calculate a new variable M1CP since the money stock is assumed to follow the partial adjustment process. The variable M1CP is then used to estimate the coefficient of adjustment. All variables are transformed using the natural logarithm with a DATA step. Refer to Balke and Gordon (1986) for data description.

The first eight observations are printed using the PRINT procedure and are shown in Output 12.5.1. Note that the first observation of the variables M1CP and INFR are missing. Therefore, the money demand equation is estimated for the period 1968:2 to 1983:4 since PROC AUTOREG ignores the first missing observation.

```
data money;
   date = intnx( 'qtr', '01jan1968'd, _n_-1 );
   format date yyqc6.;
   input m1 gnp gdf ycb @@;
   m = log( 100 * m1 / gdf );
   m1cp = log( 100 * lag(m1) / gdf );
   y = log( gnp );
   intr = log( ycb );
   infr = 100 * log( gdf / lag(gdf) );
   label m    = 'Real Money Stock (M1)'
```

```
            m1cp = 'Lagged M1/Current GDF'
            y    = 'Real GNP'
            intr = 'Yield on Corporate Bonds'
            infr = 'Rate of Prices Changes';
   datalines;
   ;
```

Output 12.5.1. Money Demand Data Series – First 8 Observations

```
Obs   date   m1      gnp     gdf   ycb    m       m1cp      y       intr    infr

  1 1968:1 187.15 1036.22 81.18 6.84 5.44041  .      6.94333 1.92279  .
  2 1968:2 190.63 1056.02 82.12 6.97 5.44732 5.42890 6.96226 1.94162 1.15127
  3 1968:3 194.30 1068.72 82.80 6.98 5.45815 5.43908 6.97422 1.94305 0.82465
  4 1968:4 198.55 1071.28 84.04 6.84 5.46492 5.44328 6.97661 1.92279 1.48648
  5 1969:1 201.73 1084.15 84.97 7.32 5.46980 5.45391 6.98855 1.99061 1.10054
  6 1969:2 203.18 1088.73 86.10 7.54 5.46375 5.45659 6.99277 2.02022 1.32112
  7 1969:3 204.18 1091.90 87.49 7.70 5.45265 5.44774 6.99567 2.04122 1.60151
  8 1969:4 206.10 1085.53 88.62 8.22 5.44917 5.43981 6.98982 2.10657 1.28331
```

The money demand equation is first estimated using OLS. The DW=4 option produces generalized Durbin-Watson statistics up to the fourth order. Their exact marginal probabilities (*p*-values) are also calculated with the DWPROB option. The Durbin-Watson test indicates positive first-order autocorrelation at, say, the 10% confidence level. You can use the Durbin-Watson table, which is available only for 1% and 5% significance points. The relevant upper (d_U) and lower (d_L) bounds are $d_U = 1.731$ and $d_L = 1.471$, respectively, at 5% significance level. However, the bounds test is inconvenient since sometimes you may get the statistic in the inconclusive region while the interval between the upper and lower bounds becomes smaller with the increasing sample size.

```
   title 'Partial Adjustment Money Demand Equation';
   title2 'Quarterly Data - 1968:2 to 1983:4';

   proc autoreg data=money outest=est covout;
      model m = m1cp y intr infr / dw=4 dwprob;
   run;
```

Output 12.5.2. OLS Estimation of the Partial Adjustment Money Demand Equation

```
                     Partial Adjustment Money Demand Equation
                        Quarterly Data - 1968:2 to 1983:4

                              The AUTOREG Procedure

                   Dependent Variable                        m
                                        Real Money Stock (M1)


                          Ordinary Least Squares Estimates

          SSE                0.00271902    DFE                      58
          MSE                 0.0000469    Root MSE            0.00685
          SBC                -433.68709    AIC              -444.40276
          Regress R-Square       0.9546    Total R-Square       0.9546


                             Durbin-Watson Statistics

                    Order            DW    Pr < DW    Pr > DW

                        1        1.7355     0.0607     0.9393
                        2        2.1058     0.5519     0.4481
                        3        2.0286     0.5002     0.4998
                        4        2.2835     0.8880     0.1120


                             Standard            Approx
Variable      DF  Estimate     Error t Value Pr > |t| Variable Label

Intercept      1    0.3084    0.2359    1.31   0.1963
m1cp           1    0.8952    0.0439   20.38   <.0001 Lagged M1/Current GDF
y              1    0.0476    0.0122    3.89   0.0003 Real GNP
intr           1   -0.0238  0.007933   -3.00   0.0040 Yield on Corporate Bonds
infr           1 -0.005646  0.001584   -3.56   0.0007 Rate of Prices Changes
```

The autoregressive model is estimated using the maximum likelihood method. Though the Durbin-Watson test statistic is calculated after correcting the autocorrelation, it should be used with care since the test based on this statistic is not justified theoretically.

```
   proc autoreg data=money;
      model m = m1cp y intr infr / nlag=1 method=ml maxit=50;
      output out=a p=p pm=pm r=r rm=rm ucl=ucl lcl=lcl
                   uclm=uclm lclm=lclm;
   run;

   proc print data=a(obs=8);
      var p pm r rm ucl lcl uclm lclm;
   run;
```

A difference is shown between the OLS estimates in Output 12.5.2 and the AR(1)-ML estimates in Output 12.5.3. The estimated autocorrelation coefficient is significantly negative (-0.88345). Note that the negative coefficient of A(1) should be interpreted as a positive autocorrelation.

Two predicted values are produced dash predicted values computed for the structural model and predicted values computed for the full model. The full model includes both the structural and error-process parts. The predicted values and residuals are stored in the output data set A, as are the upper and lower 95% confidence limits for the predicted values. Part of the data set A is shown in Output 12.5.4. The first observation is missing since the explanatory variables, M1CP and INFR, are missing for the corresponding observation.

Output 12.5.3. Estimated Partial Adjustment Money Demand Equation

```
                    Partial Adjustment Money Demand Equation
                       Quarterly Data - 1968:2 to 1983:4

                             The AUTOREG Procedure

                     Estimates of Autoregressive Parameters

                                             Standard
                   Lag      Coefficient         Error    t Value

                     1        -0.126273      0.131393      -0.96


        Algorithm converged.


                          Maximum Likelihood Estimates

         SSE                  0.00226719    DFE                     57
         MSE                   0.0000398    Root MSE           0.00631
         SBC                  -439.47665    AIC             -452.33545
         Regress R-Square         0.6954    Total R-Square      0.9621
         Durbin-Watson            2.1778


                            Standard           Approx
Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

Intercept     1    2.4121    0.4880    4.94   <.0001
m1cp          1    0.4086    0.0908    4.50   <.0001 Lagged M1/Current GDF
y             1    0.1509    0.0411    3.67   0.0005 Real GNP
intr          1   -0.1101    0.0159   -6.92   <.0001 Yield on Corporate Bonds
infr          1 -0.006348  0.001834   -3.46   0.0010 Rate of Prices Changes
AR1           1   -0.8835    0.0686  -12.89   <.0001


                   Autoregressive parameters assumed given.

                            Standard           Approx
Variable     DF  Estimate     Error t Value Pr > |t| Variable Label

Intercept     1    2.4121    0.4685    5.15   <.0001
m1cp          1    0.4086    0.0840    4.87   <.0001 Lagged M1/Current GDF
y             1    0.1509    0.0402    3.75   0.0004 Real GNP
intr          1   -0.1101    0.0155   -7.08   <.0001 Yield on Corporate Bonds
infr          1 -0.006348  0.001828   -3.47   0.0010 Rate of Prices Changes
```

Output 12.5.4. Partial List of the Predicted Values

```
  Obs    p       pm             r      rm      ucl     lcl     uclm    lclm

    1  .       .       .            .        .       .       .       .
    2 5.45962 5.45962 -.005763043 -0.012301 5.49319 5.42606 5.47962 5.43962
    3 5.45663 5.46750 0.001511258 -0.009356 5.47987 5.43340 5.48700 5.44800
    4 5.45934 5.46761 0.005574104 -0.002691 5.48267 5.43601 5.48723 5.44799
    5 5.46636 5.46874 0.003442075  0.001064 5.48903 5.44369 5.48757 5.44991
    6 5.46675 5.46581 -.002994443 -0.002054 5.48925 5.44424 5.48444 5.44718
    7 5.45672 5.45854 -.004074196 -0.005889 5.47882 5.43462 5.47667 5.44040
    8 5.44404 5.44924 0.005136019 -0.000066 5.46604 5.42203 5.46726 5.43122
```

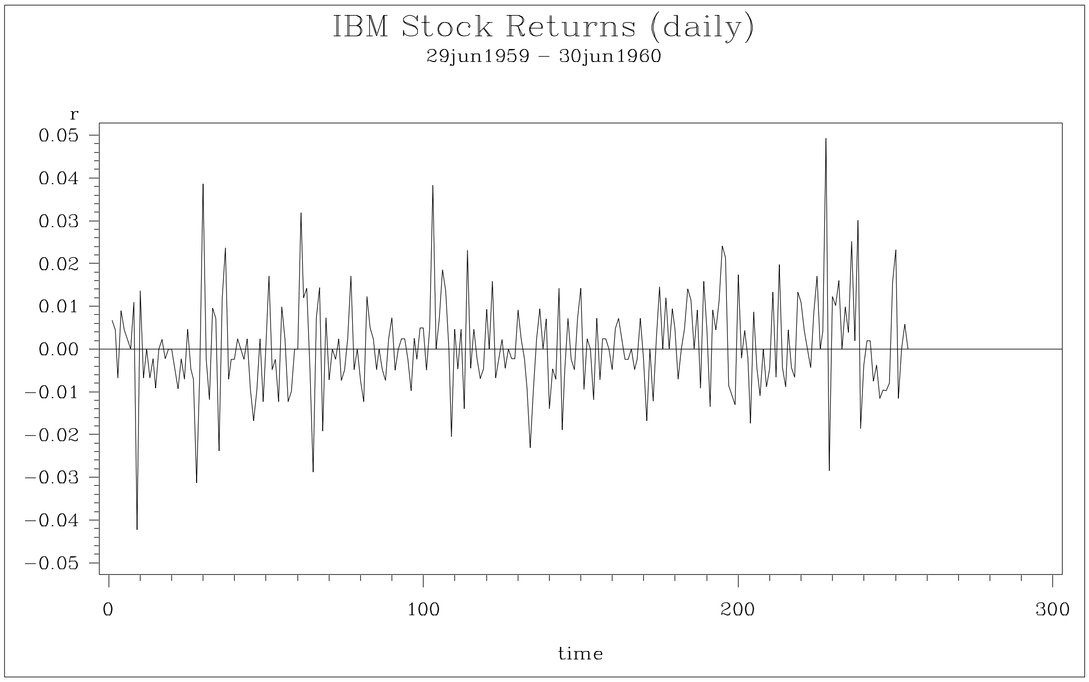
Example 12.6. Estimation of ARCH(2) Process

Stock returns show a tendency for small changes to be followed by small changes while large changes are followed by large changes. The plot of daily price changes of the IBM common stock (Box and Jenkins 1976, p 527) are shown in Output 12.6.1. The time series look serially uncorrelated, but the plot makes us skeptical of their independence.

With a DATA step, the stock (capital) returns are computed from the closing prices. To forecast the conditional variance, an additional 46 observations with missing values are generated.

```
title 'IBM Stock Returns (daily)';
title2 '29jun1959 - 30jun1960';

data ibm;
   infile datalines eof=last;
   input x @@;
   r = dif( log( x ) );
   time = _n_-1;
   output;
   return;
last:
   do i = 1 to 46;
      r = .;
      time + 1;
      output;
   end;
   return;
datalines;
;

proc gplot data=ibm;
   plot r*time / vref=0;
   symbol1 i=join v=none;
run;
```

Output 12.6.1. IBM Stock Returns: Daily



The simple ARCH(2) model is estimated using the AUTOREG procedure. The MODEL statement option GARCH=(Q=2) specifies the ARCH(2) model. The OUTPUT statement with the CEV= option produces the conditional variances V. The conditional variance and its forecast is calculated using parameter estimates:

$$h_t = \hat{\omega} + \hat{\alpha}_1 \epsilon_{t-1}^2 + \hat{\alpha}_2 \epsilon_{t-2}^2$$

$$E(\epsilon_{t+d}^2 | \Psi_t) = \hat{\omega} + \sum_{i=1}^{2} \hat{\alpha}_i E(\epsilon_{t+d-i}^2 | \Psi_t)$$

where $d > 1$.

```
proc autoreg data=ibm maxit=50;
   model r = / noint garch=(q=2);
   output out=a cev=v;
run;
```

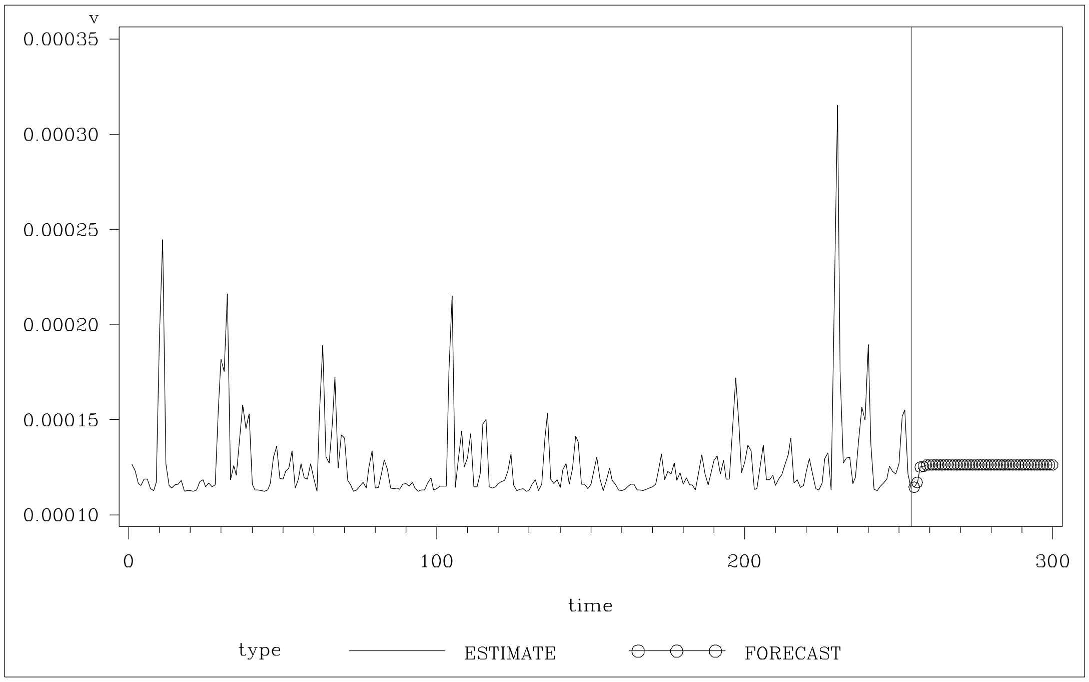
The parameter estimates for ω, α_1, and α_2 are 0.00011, 0.04136, and 0.06976, respectively. The normality test indicates that the conditional normal distribution may not fully explain the leptokurtosis in the stock returns (Bollerslev 1987).

The ARCH model estimates are shown in Output 12.6.2, and conditional variances are also shown in Output 12.6.3.

Output 12.6.2. ARCH(2) Estimation Results

```
                              The AUTOREG Procedure

                             Dependent Variable    r


                         Ordinary Least Squares Estimates

          SSE                0.03214307    DFE                      254
          MSE                 0.0001265    Root MSE             0.01125
          SBC                 -1558.802    AIC                -1558.802
          Regress R-Square       0.0000    Total R-Square        0.0000
          Durbin-Watson          2.1377

            NOTE: No intercept term is used. R-squares are redefined.


         Algorithm converged.


                                 GARCH Estimates

           SSE               0.03214307    Observations             254
           MSE                0.0001265    Uncond Var        0.00012632
           Log Likelihood    781.017441    Total R-Square        0.0000
           SBC               -1545.4229    AIC               -1556.0349
           Normality Test      105.8557    Pr > ChiSq            <.0001

            NOTE: No intercept term is used. R-squares are redefined.


                                                Standard                 Approx
       Variable        DF       Estimate           Error    t Value    Pr > |t|

       ARCH0            1       0.000112       7.5608E-6      14.85      <.0001
       ARCH1            1         0.0413          0.0511       0.81      0.4181
       ARCH2            1         0.0697          0.0432       1.62      0.1062
```

Output 12.6.3. Conditional Variance for IBM Stock Prices



Example 12.7. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics. This is a continuation of "Forecasting Autoregressive Error Models" in the section "Getting Started" on page 490. ODS graphics enables you to generate graphics similar to Figure 12.6 without resorting to the GPLOT procedure.

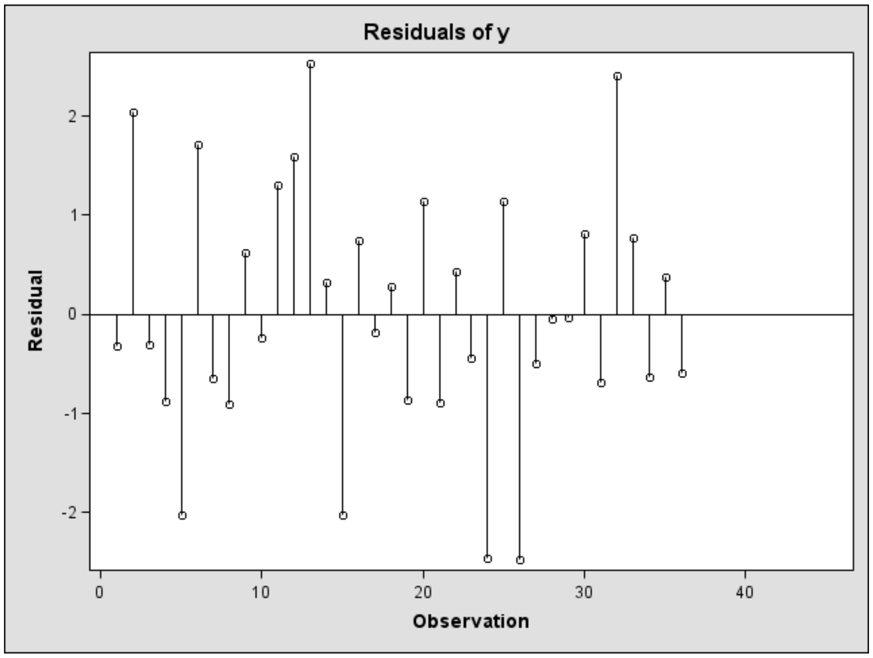
These graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the AUTOREG procedure, see the "ODS Graphics" section on page 560.

The following statements show how to generate ODS graphics plots with the AUTOREG procedure. In this case, all plots are requested using the ALL option in the PROC AUTOREG statement, in addition to the ODS GRAPHICS statement. The plots are displayed in Output 12.7.1 through Output 12.7.8.

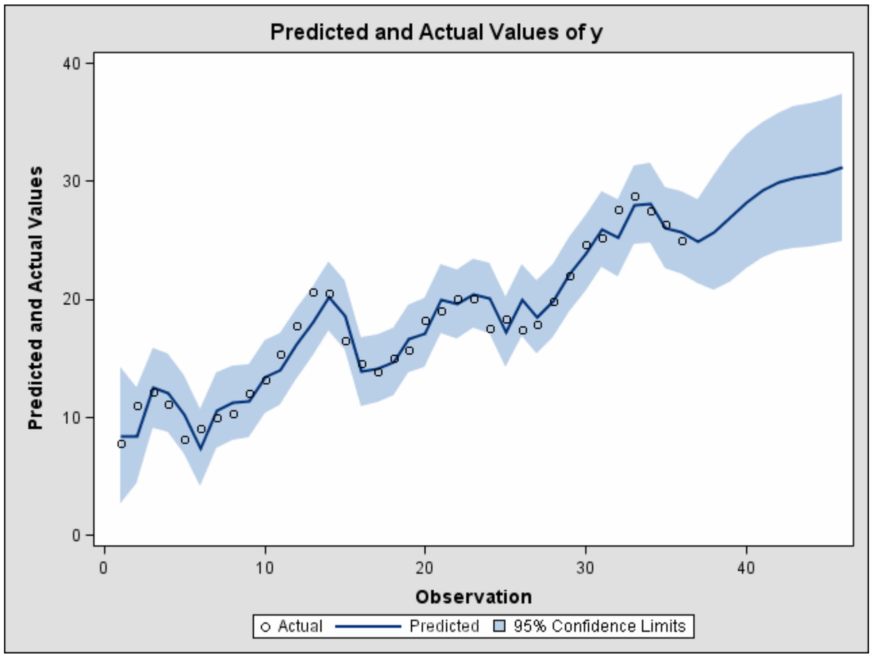
```
ods html;
ods graphics on;

proc autoreg data=b all;
   model y = time / nlag=2 method=ml;
   output out=p p=yhat pm=ytrend
             lcl=lcl ucl=ucl;
run;
```

```
ods graphics off;
ods html close;
```

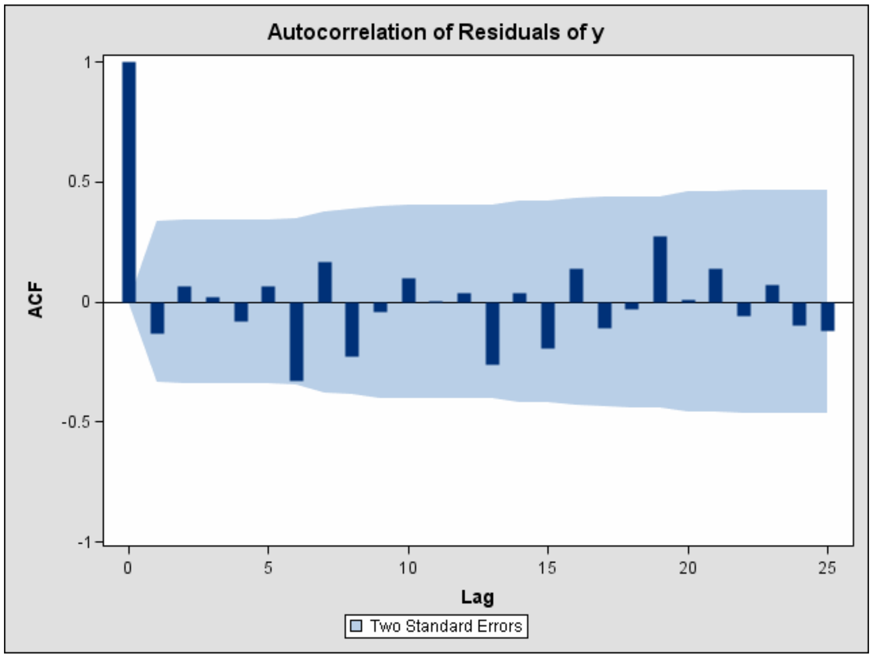
Output 12.7.1. Residuals Plot (Experimental)



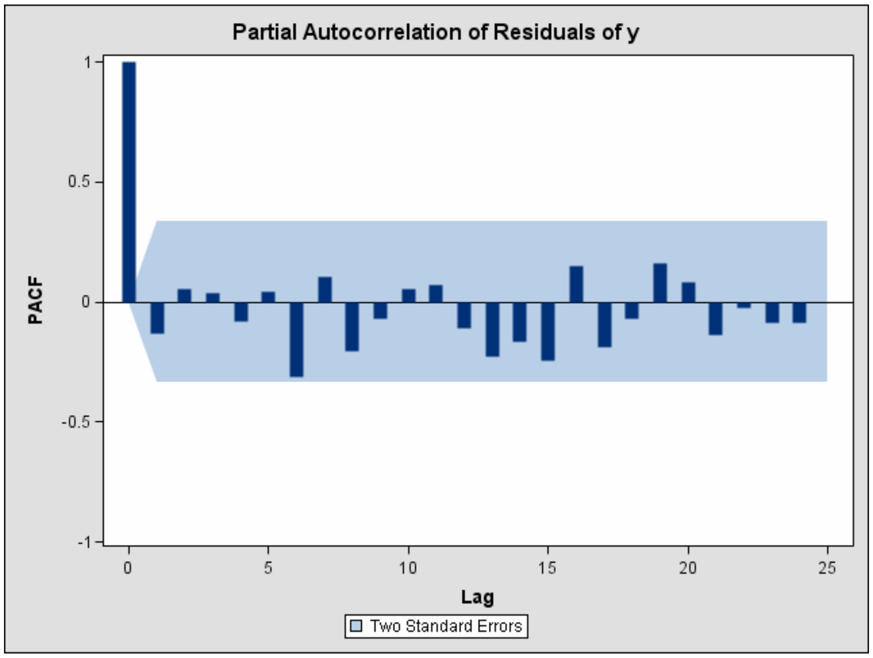
Output 12.7.2. Predicted vs Actual Plot (Experimental)



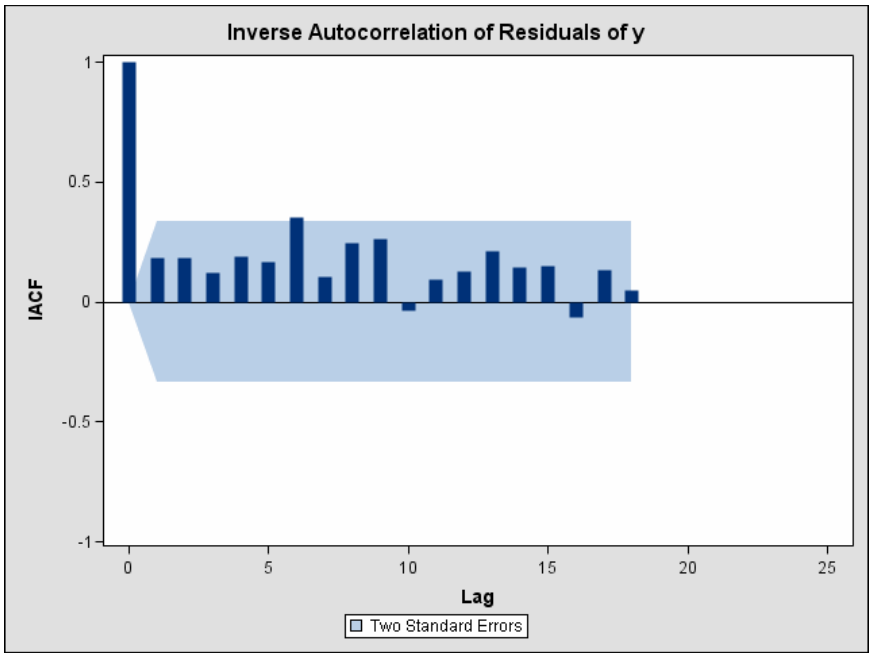
Output 12.7.3. Autocorrelation of Residuals Plot (Experimental)



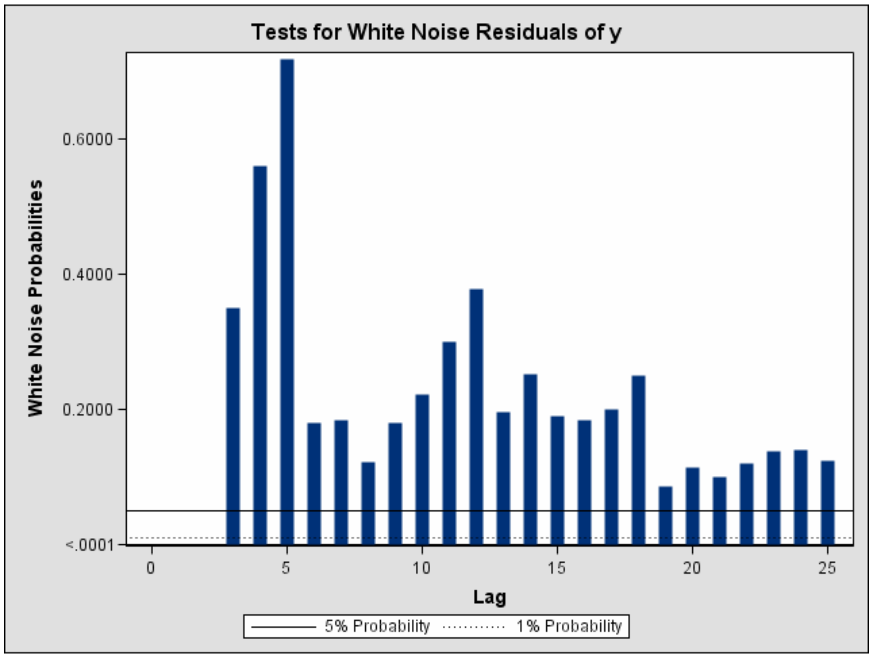
Output 12.7.4. Partial Autocorrelation of Residuals Plot (Experimental)



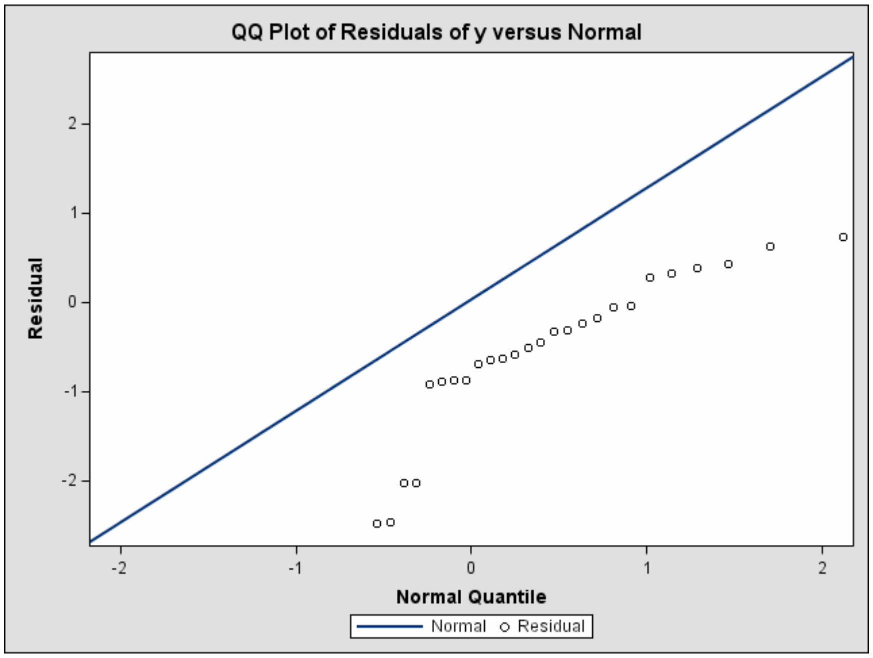
Output 12.7.5. Inverse Autocorrelation of Residuals Plot (Experimental)



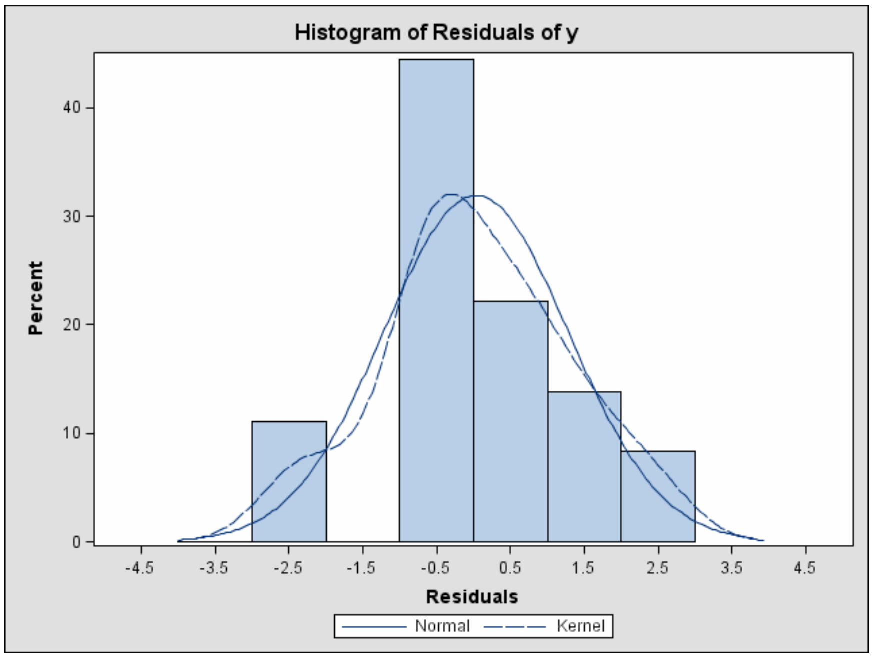
Output 12.7.6. Tests for White Noise Residuals Plot (Experimental)



Output 12.7.7. QQ Plot of Residuals (Experimental)



Output 12.7.8. Histogram of Residuals (Experimental)



References

Anderson, T.W. and Mentz, R.P. (1980), “On the Structure of the Likelihood Function of Autoregressive and Moving Average Models,” *Journal of Time Series*, 1, 83–94.

Ansley, C.F., Kohn, R., and Shively, T.S. (1992), “Computing *p*-values for the Generalized Durbin-Watson and Other Invariant Test Statistics,” *Journal of Econometrics*, 54, 277–300.

Baillie, R.T. and Bollerslev, T. (1992), “Prediction in Dynamic Models with Time-Dependent Conditional Variances,” *Journal of Econometrics*, 52, 91–113.

Balke, N.S. and Gordon, R.J. (1986) “Historical Data,” in *The American Business Cycle*, ed. R.J. Gordon, Chicago: The University of Chicago Press, 781–850.

Beach, C.M. and MacKinnon, J.G. (1978), “A Maximum Likelihood Procedure for Regression with Autocorrelated Errors,” *Econometrica*, 46, 51–58.

Bollerslev, T. (1986), “Generalized Autoregressive Conditional Heteroskedasticity,” *Journal of Econometrics*, 31, 307–327.

Bollerslev, T. (1987), “A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return,” *The Review of Economics and Statistics*, 69, 542–547.

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control*, Revised Edition, San Francisco: Holden-Day.

Breusch, T.S. and Pagan, A.R. (1979), "A Simple Test for Heteroscedasticity and Random Coefficient Variation," *Econometrica*, 47, 1287–1294.

Chipman, J.S. (1979), "Efficiency of Least Squares Estimation of Linear Trend When Residuals Are Autocorrelated," *Econometrica*, 47, 115–128.

Chow, G. (1960), "Tests of Equality between Sets of Coefficients in Two Linear Regressions," *Econometrica*, 28, 531–534.

Cochrane, D. and Orcutt, G.H. (1949), "Application of Least Squares Regression to Relationships Containing Autocorrelated Error Terms," *Journal of the American Statistical Association*, 44, 32–61.

Davies, R.B. (1973), "Numerical Inversion of a Characteristic Function," *Biometrika*, 60, 415–417.

Duffie, Darrell (1989), *Futures Markets*, Englewood Cliffs: Prentice Hall.

Durbin, J. (1969), "Tests for Serial Correlation in Regression Analysis Based on the Periodogram of Least-Squares Residuals," *Biometrika*, 56, 1–15.

Durbin, J. (1970), "Testing for Serial Correlation in Least-Squares Regression When Some of the Regressors Are Lagged Dependent Variables," *Econometrica*, 38, 410–421.

Edgerton, D. and Wells, C. (1994), "Critical Values for the CUSUMSQ Statistic in Medium and Large Sized Samples," *Oxford Bulletin of Economics and Statistics*, 56, 355–365.

Engle, R.F. (1982), "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation," *Econometrica*, 50, 987–1007.

Engle, R.F. and Bollerslev, T. (1986), "Modelling the Persistence of Conditional Variances," *Econometric Review*, 5, 1–50.

Engle, R.F.; Lilien, D.M.; and Robins, R.P. (1987), "Estimating Time Varying Risk in the Term Structure: The ARCH-M Model," *Econometrica*, 55, 391–407.

Fuller, W.A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons, Inc.

Fuller, W. (1978), *Introduction to Time Series*, New York: John Wiley & Sons, Inc.

Gallant, A.R. and Goebel, J.J. (1976), "Nonlinear Regression with Autoregressive Errors," *Journal of the American Statistical Association*, 71, 961–967.

Godfrey, L.G. (1978a), "Testing against General Autoregressive and Moving Average Error Models When the Regressors Include Lagged Dependent Variables," *Econometrica*, 46, 1293–1301.

Godfrey, L.G. (1978b), "Testing for Higher Order Serial Correlation in Regression Equations When the the Regressors Include Lagged Dependent Variables," *Econometrica*, 46, 1303–1310.

Godfrey, L.G. (1988), *Misspecification Tests in Econometrics*, New York: Cambridge University Press.

Golub, G.H. and Loan, C.F. (1989), *Matrix Computations*, Baltimore: The Johns Hopkins University Press.

Greene, W.H. (1993), *Econometric Analysis*, Second Edition, New York: Macmillan Publishing Company, Inc.

Hamilton, J.D. (1994), *Time Series Analysis*, Princeton: Princeton University Press.

Harvey, A.C. (1981), *The Econometric Analysis of Time Series*, New York: John Wiley & Sons, Inc., Chapter 6.

Harvey, A. (1990), *The Econometric Analysis of Time Series*, Second Edition, Cambridge: MIT Press.

Harvey, A.C. and McAvinchey, I.D. (1978), "The Small Sample Efficiency of Two-Step Estimators in Regression Models with Autoregressive Disturbances," Discussion Paper No. 78-10, University of British Columbia, April, 1978.

Harvey, A.C. and Phillips, G.D.A. (1979), "Maximum Likelihood Estimation of Regression Models with Autoregressive-Moving Average Disturbances," *Biometrika*, 66, 49–58.

Hildreth, C. and Lu, J.Y. (1960), "Demand Relations with Autocorrelated Disturbances," Michigan State University Agricultural Experiment Station Technical Bulletin 276, East Lansing, MI.

Inder, B.A. (1984), "Finite-Sample Power of Tests for Autocorrelation in Models Containing Lagged Dependent Variables," *Economics Letters*, 14, 179–185.

Inder, B.A. (1986), "An Approximation to the Null Distribution of the Durbin-Watson Statistic in Models Containing Lagged Dependent Variables," *Econometric Theory*, 2, 413–428.

Jarque, C.M. and Bera, A.K. (1980), "Efficient Tests for Normality, Homoskedasticity and Serial Independence of Regression Residuals," Economics Letters, 6, 255-259.

Johnston, J. (1972), *Econometric Methods*, Second Edition, New York: McGraw-Hill, Inc.

Jones, R.H. (1980), "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations," *Technometrics*, 22, 389–395.

Judge, G.G.; Griffiths, W.E.; Hill, R.C.; and Lee, T.C. (1985), *The Theory and Practice of Econometrics*, Second Edition, New York: John Wiley & Sons, Inc.

King, M.L. and Wu, P.X. (1991), "Small-Disturbance Asymptotics and the Durbin-Watson and Related Tests in the Dynamic Regression Model," *Journal of Econometrics*, 47, 145–152.

L'Esperance, W.L.; Chall, D.; and Taylor, D. (1976), "An Algorithm for Determining the Distribution Function of the Durbin-Watson Test Statistic," *Econometrica*, 44, 1325–1326.

Maddala, G.S. (1977), *Econometrics*, New York: McGraw-Hill, 280–281.

Maeshiro, A. (1976), "Autoregressive Transformation, Trended Independent Variables and Autocorrelated Disturbance Terms," *Review of Economics and Statistics*, 58, 497–500.

McLeod, A.I. and Li, W.K. (1983), "Diagnostic Checking ARMA Time Series Models Using Squared-Residual Autocorrelations," *Journal of Time Series Analysis*, 4, 269–273.

Nelson, C.R. and Plosser, C.I. (1982), "Trends and Random Walks in Macroeconomic Time Series: Some Evidence and Implications." *Journal of Monetary Economics*, 10, 139–162.

Nelson, D.B. (1990), "Stationarity and Persistence in the GARCH(1,1) Model," *Econometric Theory*, 6, 318–334.

Nelson, D.B. (1991), "Conditional Heteroskedasticity in Asset Returns: A New Approach," *Econometrica*, 59, 347–370.

Nelson, D.B. and Cao, C.Q. (1992), "Inequality Constraints in the Univariate GARCH Model," *Journal of Business & Economic Statistics*, 10, 229–235.

Park, R.E. and Mitchell, B.M. (1980), "Estimating the Autocorrelated Error Model with Trended Data," *Journal of Econometrics*, 13, 185–201.

Phillips, P.C. and Perron, P. (1988), "Testing for a Unit Root in Time Series Regression," *Biometrika*, 75, 335–346.

Phillips, P.C. and Ouliaris, S. (1990), "Asymptotic Properties of Residual Based Tests for Cointegration," *Econometrica*, 58, 165–193.

Prais, S.J. and Winsten, C.B. (1954), "Trend Estimators and Serial Correlation," Cowles Commission Discussion Paper No. 383.

Ramsey, J.B. (1969), "Tests for Specification Errors in Classical Linear Least Squares Regression Analysis," *Journal of Royal Statistical Society, Series B*, 31, 350–371.

Savin, N.E. and White, K.J. (1978), "Testing for Autocorrelation with Missing Observations," *Econometrica*, 46, 59–66.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461–464.

Shively, T.S. (1990), "Fast Evaluation of the Distribution of the Durbin-Watson and Other Invariant Test Statistics in Time Series Regression," *Journal of the American Statistical Association*, 85, 676–685.

Spitzer, John J. (1979), "Small-Sample Properties of Nonlinear Least Squares and Maximum Likelihood Estimators in the Context of Autocorrelated Errors," *Journal of the American Statistical Association*, 74, 41–47.

Theil, H. (1971), *Principles of Econometrics*, New York: John Wiley & Sons, Inc.

Vinod, H.D. (1973), "Generalization of the Durbin-Watson Statistic for Higher Order Autoregressive Process," it Communications in Statistics, 2, 115–144.

Wallis, K.F. (1972), "Testing for Fourth Order Autocorrelation in Quarterly Regression Equations," *Econometrica*, 40, 617–636.

White, H. (1982), "Maximum Likelihood Estimation of Misspecified Models," *Econometrica*, 50, 1–25.

White, K.J. (1992), "The Durbin-Watson Test for Autocorrelation in Nonlinear Models," *Review of Economics and Statistics*, 74, 370–373.

Chapter 13

The COMPUTAB Procedure

Chapter Contents

Chapter 13
The COMPUTAB Procedure

Overview

The COMPUTAB procedure (**COMPU**ting and **TAB**ular reporting) produces tabular reports generated using a programmable data table.

The COMPUTAB procedure is especially useful when you need both the power of a programmable spreadsheet and a report generation system, but you want to set up a program to run in a batch mode and generate routine reports.

With PROC COMPUTAB, you can select a subset of observations from the input data set, define the format of a table, operate on its row and column values, and create new columns and rows. Access to individual table values is available when needed.

The COMPUTAB procedure can tailor reports to almost any desired specification and provide consolidation reports over summarization variables. The generated report values can be stored in an output data set. It is especially useful in creating tabular reports such as income statements, balance sheets, and other row and column reports.

Getting Started

The following example shows the different types of reports that can be generated by PROC COMPUTAB.

Suppose a company has monthly expense data on three of its divisions and wants to produce the year-to-date expense report shown in Figure 13.1. This section starts out with the default report produced by the COMPUTAB procedure and modifies it until the desired report is achieved.

```
                             Year to Date Expenses

                                   Division  Division  Division          All
                                          A         B         C    Divisions
       Travel Expenses within U.S.    18700    211000     12800     $242,500
       Advertising                    18500    176000     34500     $229,000
       Permanent Staff Salaries      186000   1270000    201000   $1,657,000
       Benefits Including Insurance    3900     11100     17500      $32,500
                                   ========  ========  ========   ==========
       Total                         227100   1668100    265800   $2,161,000
```

Figure 13.1. Year to Date Expense Report

Producing a Simple Report

Without any specifications, the COMPUTAB procedure transposes and prints the input data set. The variables in the input data set become rows in the report, and the observations in the input data set become columns. The variable names are used as the row titles. The column headings default to COL1 through COLn. For example, the following input data set contains the monthly expenses reported by different divisions of the company:

```
data report;
   input compdiv $ date:date7. salary travel insure advrtise;
   format date date7.;
   label travel   = 'Travel Expenses within U.S.'
         advrtise = 'Advertising'
         salary   = 'Permanent Staff Salaries'
         insure   = 'Benefits Including Insurance';
   datalines;
A 31JAN1989 95000  10500  2000 6500
B 31JAN1989 668000 112000 5600 90000
C 31JAN1989 105000 6800   9000 18500
A 28FEB1989 91000  8200   1900 12000
B 28FEB1989 602000 99000  5500 86000
C 28FEB1989 96000  6000   8500 16000
;
```

You can get a listing of the data set by using the PRINT procedure, as follows:

```
title 'Listing of Monthly Divisional Expense Data';
proc print data=report;
run;
```

```
                  Listing of Monthly Divisional Expense Data

    Obs    compdiv         date     salary    travel    insure    advrtise

     1        A       31JAN1989      95000     10500     2000        6500
     2        B       31JAN1989     668000    112000     5600       90000
     3        C       31JAN1989     105000      6800     9000       18500
     4        A       28FEB1989      91000      8200     1900       12000
     5        B       28FEB1989     602000     99000     5500       86000
     6        C       28FEB1989      96000      6000     8500       16000
```

Figure 13.2. Listing of Data Set by PROC PRINT

To get a simple, transposed report of the same data set, use the following PROC COMPUTAB statement:

```
title 'Monthly Divisional Expense Report';
proc computab data=report;
run;
```

```
                        Monthly Divisional Expense Report

                 COL1        COL2        COL3        COL4        COL5        COL6

compdiv             A           B           C           A           B           C
date        31JAN1989   31JAN1989   31JAN1989   28FEB1989   28FEB1989   28FEB1989
salary       95000.00   668000.00   105000.00    91000.00   602000.00    96000.00
travel       10500.00   112000.00     6800.00     8200.00    99000.00     6000.00
insure        2000.00     5600.00     9000.00     1900.00     5500.00     8500.00
advrtise      6500.00    90000.00    18500.00    12000.00    86000.00    16000.00
```

Figure 13.3. Listing of Data Set by PROC COMPUTAB

Using PROC COMPUTAB

The COMPUTAB procedure is best understood by examining the following features:

- definition of the report layout with ROWS and COLUMNS statements
- input block
- row blocks
- column blocks

PROC COMPUTAB builds a table according to the specifications in the ROWS and COLUMNS statements. Row names and column names define the rows and columns of the table. Options in the ROWS and COLUMNS statements control titles, spacing, and formatting.

The input block places input observations into the appropriate columns of the report. It consists of programming statements used to select observations to be included in the report, to determine the column into which the observation should be placed, and to calculate row and column values that are not in the input data set.

Row blocks and column blocks perform operations on the values of rows and columns of the report after the input block has executed. Row blocks are a block of programming statements labeled ROW*xxxxx*: that create or modify row values; column blocks are a block of programming statements labeled COL*xxxxx*: that create or modify column values. Row and column blocks can make multiple passes through the report for final calculations.

For most reports, these features are sufficient. More complicated applications may require knowledge of the program data vector and the COMPUTAB data table. These topics are discussed in the section "Details" later in this chapter.

Defining Report Layout

ROWS and COLUMNS statements define the rows and columns of the report. The order of row and column names on these statements determines the order of rows and columns in the report. Additional ROWS and COLUMNS statements can be used to specify row and column formatting options.

The following statements select and order the variables from the input data set and produce the report in Figure 13.4:

```
proc computab data=report;
   rows travel advrtise salary;
run;
```

```
                   COL1        COL2        COL3        COL4        COL5        COL6

TRAVEL         10500.00   112000.00     6800.00     8200.00    99000.00     6000.00
ADVRTISE        6500.00    90000.00    18500.00    12000.00    86000.00    16000.00
SALARY         95000.00   668000.00   105000.00    91000.00   602000.00    96000.00
INSURE          2000.00     5600.00     9000.00     1900.00     5500.00     8500.00
```

Figure 13.4. Report Produced Using a ROWS Statement

When a COLUMNS statement is not specified, each observation becomes a new column. If you use a COLUMNS statement, you must specify to which column each observation belongs by using program statements for column selection. When more than one observation is selected for the same column, values are summed.

The following statements produce Figure 13.5:

```
proc computab data= report;
   rows travel advrtise salary insure;
   columns a b c;
   *----select column for company division,
        based on value of compdiv----*;
   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';
run;
```

The statement A=COMPDIV='A'; illustrates the use of logical operators as a selection technique. If COMPDIV='A', then the current observation is added to the A column. Refer to *SAS Language: Reference, Version 6, First Edition* for more information on logical operators.

```
                            A           B           C

            TRAVEL     18700.00  211000.00   12800.00
            ADVRTISE   18500.00  176000.00   34500.00
            SALARY    186000.00  1270000.0  201000.00
            INSURE      3900.00   11100.00   17500.00
```

Figure 13.5. Report Produced Using ROWS and COLUMNS Statements

Adding Computed Rows and Columns

In addition to the variables and observations in the input data set, you can create additional rows or columns by using SAS programming statements in PROC COMPUTAB. You can

- modify input data and select columns in the input block
- create or modify columns in column blocks
- create or modify rows in row blocks

The following statements add one computed row (SUM) and one computed column (TOTAL) to the report in Figure 13.5. In the input block the logical operators indicate the observations corresponding to each column of the report. After the input block reads in the values from the input data set, the column block creates the column variable TOTAL by summing the columns A, B, and C. The additional row variable, SUM, is calculated as the sum of the other rows. The result is shown in Figure 13.6.

```
proc computab data= report;
   rows travel advrtise salary insure sum;
   columns a b c total;
   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';
   colblk: total = a + b + c;
   rowblk: sum   = travel + advrtise + salary + insure;
run;
```

```
                      A           B           C       TOTAL

      TRAVEL     18700.00  211000.00   12800.00  242500.00
      ADVRTISE   18500.00  176000.00   34500.00  229000.00
      SALARY    186000.00  1270000.0  201000.00  1657000.0
      INSURE      3900.00   11100.00   17500.00   32500.00
      SUM       227100.00  1668100.0  265800.00  2161000.0
```

Figure 13.6. Report Produced Using Row and Column Blocks

Enhancing the Report

To enhance the appearance of the final report, you can use

- TITLE and LABEL statements
- column headings
- row titles
- row and column spacing control
- overlining and underlining
- formats

The following example enhances the report in the previous example. The enhanced report is shown in Figure 13.7.

The TITLE statement assigns the report title. The column headings in Figure 13.7 (Division A, Division B, and Division C) are assigned in the first COLUMNS statement by “Division” _name_ specification. The second COLUMNS statement assigns the column heading (“All” “Divisions”), sets the spacing (+4), and formats the values in the TOTAL column.

Similarly, the first ROWS statement uses previously assigned variable labels for row labels by specifying the _LABEL_ option. The DUL option in the second ROWS statement double underlines the INSURE row. The third ROWS statement assigns the row label TOTAL to the SUM row.

```
title 'Year to Date Expenses';

proc computab cwidth=8 cdec=0;

   columns a  b  c / 'Division' _name_;
   columns total / 'All' 'Divisions' +4 f=dollar10.0;

   rows travel advrtise salary insure / _label_;
   rows insure / dul;
   rows sum / 'Total';

   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';

   colblk: total = a + b + c;
   rowblk: sum   = travel + advrtise + salary + insure;
run;
```

```
                        Year to Date Expenses

                                Division  Division  Division        All
                                       A         B         C  Divisions
   Travel Expenses within U.S.     18700    211000     12800   $242,500
   Advertising                     18500    176000     34500   $229,000
   Permanent Staff Salaries       186000   1270000    201000 $1,657,000
   Benefits Including Insurance     3900     11100     17500    $32,500
                                ========  ========  ======== ==========
   Total                          227100   1668100    265800 $2,161,000
```

Figure 13.7. Report Produced by PROC COMPUTAB Using Enhancements

Syntax

The following statements are used with the COMPUTAB procedure:

PROC COMPUTAB *options*;
  BY *variables*;
  COLUMNS *names / options*;
  ROWS *names / options*;
  CELL *names /* **FORMAT=** *format*;
  INIT *anchor-name* ***locator-name*** *values* ***locator-name values***;
  programming *statements*;
  SUMBY *variables*;

The PROC COMPUTAB statement is the only required statement. The COLUMNS, ROWS, and CELL statements define the COMPUTAB table. The INIT statement initializes the COMPUTAB table values. Programming statements process COMPUTAB table values. The BY and SUMBY statements provide BY-group processing and consolidation (roll up) tables.

Functional Summary

COMPUTAB procedure statements and options are summarized in the following table:

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify the format for printing a particular cell	CELL	
define columns of the report	COLUMNS	
initialize values in the COMPUTAB data table	INIT	
define rows of the report	ROWS	
produce consolidation tables	SUMBY	

Description	Statement	Option
Data Set Options		
specify the input data set	COMPUTAB	DATA=
specify an output data set	COMPUTAB	OUT=
Input Options		
specify a value to use when testing for 0	COMPUTAB	FUZZ=
initialize the data table to missing	COMPUTAB	INITMISS
prevent the transposition of the input data set	COMPUTAB	NOTRANS
Printing Control Options		
suppress printing of the listed columns	COLUMNS	NOPRINT
suppress all printed output	COMPUTAB	NOPRINT
suppress printing of the listed rows	ROWS	NOPRINT
suppress columns with all 0 or missing values	COLUMNS	NOZERO
suppress rows with all 0 or missing values	ROWS	NOZERO
list option values	COMPUTAB	OPTIONS
overprint titles, values, overlining, and underlining associated with listed rows	ROWS	OVERPRINT
print only consolidation tables	COMPUTAB	SUMONLY
Report Formatting Options		
specify number of decimal places to print	COMPUTAB	CDEC=
specify number of spaces between columns	COMPUTAB	CSPACE=
specify column width for the report	COMPUTAB	CWIDTH=
overlines the listed rows with double lines	ROWS	DOL
underline the listed rows with double lines	ROWS	DUL
specify a format for printing the cell values	CELL	FORMAT=
specify a format for printing column values	COLUMNS	FORMAT=
specify a format for printing the row values	ROWS	FORMAT=
left align the column headings	COLUMNS	LJC
left-justify character rows in each column	ROWS	LJC
specify indentation from the margin	ROWS	+n
suppress printing of row titles on later pages	COMPUTAB	NORTR
overlines the listed rows with a single line	ROWS	OL
starts a new page before printing the listed rows	ROWS	_PAGE_
specify number of spaces before row titles	COMPUTAB	RTS=
print a blank row	ROWS	SKIP
underline the listed rows with a single line	ROWS	UL
specify text to print if column is 0 or missing	COLUMNS	ZERO=
specify text to print if row is 0 or missing	ROWS	ZERO=
Row and Column Type Options		

Description	Statement	Option
specify that columns contain character data	COLUMNS	CHAR
specify that rows contain character data	ROWS	CHAR
Options for Column Headings		
specify literal column headings	COLUMNS	‘column heading’
use variable labels in column headings	COLUMNS	_LABEL_
specify a master title centered over columns	COLUMNS	MTITLE=
use column names in column headings	COLUMNS	_NAME_
Options for Row Titling		
use labels in row titles	ROWS	_LABEL_
use row names in row titles	ROWS	_NAME_
specify literal row titles	ROWS	‘row title’

PROC COMPUTAB Statement

PROC COMPUTAB *options;*

The following options can be used in the PROC COMPUTAB statement:

Input Options

DATA= *SAS-data-set*
names the SAS data set containing the input data. If this option is not specified, the last created data set is used. If you are not reading a data set, use DATA=_NULL_.

FUZZ= *value*
specifies the criterion to use when testing for 0. If a number is within the FUZZ= value of 0, the number is set to 0.

INITMISS
initializes the COMPUTAB data table to missing rather than to 0. The COMPUTAB data table is discussed further in the section "Details" later in this chapter.

NOTRANSPOSE
NOTRANS
prevents the transposition of the input data set in building the COMPUTAB report tables. The NOTRANS option causes input data set variables to appear among the columns of the report rather than among the rows.

Report Formatting Options

The formatting options specify default values. Many of the formatting options can be modified for specific columns in COLUMNS statements and for rows in ROWS statements.

CDEC= *d*
specifies the default number of decimal places for printing. The default is CDEC=2. See the FORMAT= option in the sections on COLUMN, ROWS, and CELL statements later in this chapter.

CSPACE= *n*
specifies the default number of spaces to insert between columns. The value of the CSPACE= option is used as the default value for the +*n* option in the COLUMNS statement. The default is CSPACE=2.

CWIDTH= *w*
specifies a default column width for the report. The default is CWIDTH=9. The width must be in the range of 1-32.

NORTR
suppresses the printing of row titles on each page. The NORTR (no row-title repeat) option is useful to suppress row titles when report pages are to be joined together in a larger report.

RTS= *n*
specifies the default number of spaces to be inserted before row titles when row titles appear after the first printed column. The default row-title spacing is RTS=2.

Output Options

NOPRINT
suppresses all printed output. Use the NOPRINT option with the OUT= option to produce an output data set but no printed reports.

OPTIONS
lists PROC COMPUTAB option values. The option values appear on a separate page preceding the procedure's normal output.

OUT= *SAS-data-set*
names the SAS data set to contain the output data. See the section "Details" for a description of the structure of the output data set.

SUMONLY
suppresses printing of detailed reports. When the SUMONLY option is used, PROC COMPUTAB generates and prints only consolidation tables as specified in the SUMBY statement.

COLUMNS Statement

COLUMNS *column-list* **/** *options*;

COLUMNS statements define the columns of the report. The COLUMNS statement can be abbreviated COLUMN, COLS, or COL.

The specified column names must be valid SAS names. Abbreviated lists, as described in *SAS Language: Reference*, can also be used.

You can use as many COLUMNS statements as you need. A COLUMNS statement can describe more than one column, and one column of the report can be described with several different COLUMNS statements. The order of the columns on the report is determined by the order of appearance of column names in COLUMNS statements. The first occurrence of the name determines where in the sequence of columns a particular column is located.

The following options can be used in the COLUMNS statement:

Option for Column Type

CHAR
indicates that the columns contain character data.

Options for Column Headings

You can specify as many lines of column headings as needed. If no options are specified, the column names from the COLUMNS statement are used as column headings. Any or all of the following options can be used in a column heading:

“column heading”
specifies that the characters enclosed in quotes is to be used in the column heading for the variable or variables listed in the COLUMNS statement. Each quoted string appears on a separate line of the heading.

LABEL
uses labels, if provided, in the heading for the column or columns listed in the COLUMNS statement. If a label has not been provided, the name of the column is used. Refer to *SAS Language: Reference* for information on the LABEL statement.

MTITLE= *“text”*
specifies that the string of characters enclosed in quotes is a master title to be centered over all the columns listed in the COLUMNS statement. The list of columns must be consecutive. Special characters (“+”, “*”, “=”, and so forth) placed on either side of the text expand to fill the space. The MTITLE= option can be abbreviated M=.

NAME
uses column names in column headings for the columns listed in the COLUMNS statement. This option allows headings (*“text”*) and names to be combined in a heading.

Options for Column Print Control

+n
inserts *n* spaces before each column listed in the COLUMNS statement. The default spacing is given by the CSPACE= option in the PROC COMPUTAB statement.

NOPRINT
suppresses printing of columns listed in the COLUMNS statement. This option enables you to create columns to be used for intermediate calculations without having those columns printed.

NOZERO
suppresses printing of columns when all the values in a column are 0 or missing. Numbers within the FUZZ= value of 0 are treated as 0.

PAGE
starts a new page of the report before printing each of the columns in the list that follows.

TITLES
prints row titles before each column in the list. The _TITLES_ option can be abbreviated as _TITLE_.

Options for Column Formatting

Column formats override row formats for particular table cells only when the input data set is not transposed (when the NOTRANS option is specified).

FORMAT= *format*
specifies a format for printing the values of the columns listed in the COLUMNS statement. The FORMAT= option can be abbreviated F=.

LJC
left-justifies the column headings for the columns listed. By default, columns are right-justified. When the LJC (left-justify character) option is used, any character row values in the column are also left-justified rather than right-justified.

ZERO= *"text"*
substitutes *"text"* when the value in the column is 0 or missing.

ROWS Statement

ROWS *row-list* **/** *options***;**

ROWS statements define the rows of the report. The ROWS statement can be abbreviated ROW.

The specified row names must be valid SAS names. Abbreviated lists, as described in *SAS Language: Reference*, can also be used.

You can use as many ROWS statements as you need. A ROWS statement can describe more than one row, and one row of the report can be described with several different ROWS statements. The order of the rows in the report is determined by the order of appearance of row names in ROWS statements. The first occurrence of the name determines where the row is located.

The following options can be used in the ROWS statement:

Option for Row Type

CHAR
indicates that the rows contain character data.

Options for Row Titling

You can specify as many lines of row titles as needed. If no options are specified, the names from the ROWS statement are used as row titles. Any or all of the following options can be used in a row title:

LABEL
: uses labels as row titles for the row or rows listed in the ROWS statement. If a label is not provided, the name of the row is substituted. Refer to *SAS Language: Reference* for more information on the LABEL statement.

NAME
: uses row names in row titles for the row or rows listed in the ROWS statement.

"row title"
: specifies that the string of characters enclosed in quotes is to be used in the row title for the row or rows listed in the ROWS statement. Each quoted string appears on a separate line of the heading.

Options for Row Print Control

+n
: indents *n* spaces from the margin for the rows in the ROWS statement.

DOL
: overlines the rows listed in the ROWS statement with double lines. Overlines are printed on the line before any row titles or data for the row.

DUL
: underlines the rows listed in the ROWS statement with double lines. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

NOPRINT
: suppresses printing of the rows listed in the ROWS statement. This option enables you to create rows to be used for intermediate calculations without having those rows printed.

NOZERO
: suppresses the printing of a row when all the values are 0 or missing.

OL
: overlines the rows listed in the ROWS statement with a single line. Overlines are printed on the line before any row titles or data for the row.

OVERPRINT
: overprints titles, values, overlining, and underlining associated with rows listed in the ROWS statement. The OVERPRINT option can be abbreviated OVP. This option is valid only when the system option OVP is in effect. Refer to *SAS Language: Reference* for more information about the OVP option.

PAGE
starts a new page of the report before printing these rows.

SKIP
prints a blank line after the data lines for these rows.

UL
underlines the rows listed in the ROWS statement with a single line. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

Options for Row Formatting

Row formatting options take precedence over column-formatting options when the input data set is transposed. Row print width can never be wider than column width. Character values are truncated on the right.

FORMAT= *format*
specifies a format for printing the values of the rows listed in the ROWS statement. The FORMAT= option can be abbreviated as F=.

LJC
left-justifies character rows in each column.

ZERO= *“text”*
substitutes *text* when the value in the row is 0 or missing.

CELL Statement

CELL *cell_names* **/ FORMAT=** *format***;**

The CELL statement specifies the format for printing a particular cell in the COMPUTAB data table. Cell variable names are compound SAS names of the form *name1.name2*, where *name1* is the name of a row variable and *name2* is the name of a column variable. Formats specified with the FORMAT= option in CELL statements override formats specified in ROWS and COLUMNS statements.

INIT Statement

INIT *anchor-name* **[***locator-name***]** *values* **[***locator-name values***];**

The INIT statement initializes values in the COMPUTAB data table at the beginning of each execution of the procedure and at the beginning of each BY group if a BY statement is present.

The INIT statement in the COMPUTAB procedure is similar in function to the RETAIN statement in the DATA step, which initializes values in the program data vector. The INIT statement can be used at any point after the variable to which it refers has been defined in COLUMNS or ROWS statements. Each INIT statement initializes one row or column. Any number of INIT statements can be used.

The first term after the keyword INIT, *anchor-name*, anchors initialization to a row or column. If *anchor-name* is a row name, then all *locator-name* values in the statement are columns of that row. If *anchor-name* is a column name, then all *locator-name* values in the statement are rows of that column.

The following terms appear in the INIT statement:

anchor-name
: names the row or column in which values are to be initialized. This term is required.

locator-name
: identifies the starting column in the row (or starting row in the column) into which values are to be placed. For example, in a table with a row SALES and a column for each month of the year, the following statement initializes values for columns JAN, FEB, and JUN:

```
init sales jan 500 feb 600 jun 800;
```

If you do not specify *locator-name* values, the first value is placed into the first row or column, the second value into the second row or column, and so on. For example,

```
init sales 500 600 450;
```

assigns 500 to column JAN, 600 to FEB, and 450 to MAR.

+n
: specifies the number of columns in a row (or rows in a column) that are to be skipped when initializing values. For example, the statement

```
init sales jan 500 +5 900;
```

assigns 500 to JAN and 900 to JUL.

n*value
: assigns *value* to *n* columns in the row (or rows in the column). For example, the statement

```
init sales jan 6*500 jul 6*1000;
```

and the statement

```
init sales 6*500 6*1000;
```

both assign 500 to columns JAN through JUN and 1000 to JUL through DEC.

Programming Statements

You can use most SAS programming statements the same way you use them in the DATA step. Also, all DATA step functions can be used in the COMPUTAB procedure.

Lines written by the PUT statement are not integrated with the COMPUTAB report. PUT statement output is written to the SAS log.

The automatic variable _N_ can be used; its value is the number of observations read or the number read in the current BY group, if a BY statement is used. FIRST.*variable* and LAST.*variable* references cannot be used.

The following statements are also available in PROC COMPUTAB:

```
ABORT                       FORMAT
ARRAY                       GOTO
ATTRIB                      IF-THEN/ELSE
assignment statement        LABEL
CALL                        LINK
DELETE                      PUT
DO                          RETAIN
iterative DO                SELECT
DO UNTIL                    STOP
DO WHILE                    sum statement
END                         TITLE
FOOTNOTE
```

The programming statements can be assigned labels ROW*xxxxx*: or COL*xxxxx*: to indicate the start of a row and column block, respectively. Statements in a row block create or change values in all the columns in the specified rows. Similarly, statements in a column block create or change values in all the rows in the specified columns.

There is an implied RETURN statement before each new row or column block. Thus, the flow of execution does not leave the current row (column) block before the block repeats for all columns (rows.) Row and column variables and nonretained variables are initialized prior to each execution of the block.

The next COL*xxxxx*: label, ROW*xxxxx*: label, or the end of the PROC COMPUTAB step signals the end of a row (column) block. Column blocks and row blocks can be mixed in any order. In some cases, performing calculations in different orders can lead to different results.

See "Program Flow Example," "Order of Calculations," and "Controlling Execution within Row and Column Blocks" in the section "Details" for more information.

BY Statement

BY *variables;*

A BY statement can be used with PROC COMPUTAB to obtain separate reports for observations in groups defined by the BY variables. At the beginning of each BY group, before PROC COMPUTAB reads any observations, all table values are set to 0 unless the INITMISS option or an INIT statement is specified.

SUMBY Statement

SUMBY *variables;*

The SUMBY statement produces consolidation tables for variables whose names are in the SUMBY list. Only one SUMBY statement can be used.

To use a SUMBY statement, you must use a BY statement. The SUMBY and BY variables must be in the same relative order in both statements, for example:

```
by a b c;
sumby a b;
```

This SUMBY statement produces tables that consolidate over values of C within levels of B and over values of B within levels of A. Suppose A has values 1,2; B has values 1,2; and C has values 1,2,3. Table 13.1 indicates the consolidation tables produced by the SUMBY statement.

Table 13.1. Consolidation Tables Produced by the SUMBY Statement

SUMBY Consolidations	**Consolidated BY Groups**		
A=1,B=1	C=1	C=2	C=3
A=1,B=2	C=1	C=2	C=3

A=1	B=1,C=1
	B=2,C=1
		B=1,C=2
		B=2,C=2
			B=1,C=3
			B=2,C=3

A=2,B=1	C=1	C=2	C=3
A=2,B=2	C=1	C=2	C=3

A=2	B=1,C=1
	B=2,C=1
		B=1,C=2
		B=2,C=2
			B=1,C=3
			B=2,C=3

Two consolidation tables for B are produced for each value of A. The first table consolidates the three tables produced for the values of C while B is 1; the second table consolidates the three tables produced for C while B is 2.

Tables are similarly produced for values of A. Nested consolidation tables are produced for B (as described previously) for each value of A. Thus, this SUMBY statement produces a total of six consolidation tables in addition to the tables produced for each BY group.

To produce a table that consolidates the entire data set (the equivalent of using PROC COMPUTAB with neither BY nor SUMBY statements), use the special name _TOTAL_ as the first entry in the SUMBY variable list, for example,

```
sumby _total_ a b;
```

PROC COMPUTAB then produces consolidation tables for SUMBY variables as well as a consolidation table for all observations.

To produce only consolidation tables, use the SUMONLY option in the PROC COMPUTAB statement.

Details

NOTRANS Option

The NOTRANS option in the PROC COMPUTAB statement prevents the transposition of the input data set. NOTRANS affects the input block, the precedence of row and column options, and the structure of the output data set if the OUT= option is specified.

When the input data set is transposed, input variables are among the rows of the COMPUTAB report, and observations compose columns. The reverse is true if the data set is not transposed; therefore, the input block must select rows to receive data values, and input variables are among the columns.

Variables from the input data set dominate the format specification and data type. When the input data set is transposed, input variables are among the rows of the report, and row options take precedence over column options. When the input data set is not transposed, input variables are among the columns, and column options take precedence over row options.

Variables for the output data set are taken from the dimension (row or column) that contains variables from the input data set. When the input data set is transposed, this dimension is the row dimension; otherwise, the output variables come from the column dimension.

Program Flow Example

This example shows how the COMPUTAB procedure processes observations in the program working storage and the COMPUTAB data table (CDT).

Assume you have three years of sales and cost of goods sold (CGS) figures, and you want to determine total sales and cost of goods sold and calculate gross profit and the profit margin.

```
data example;
   input year sales cgs;
   datalines;
1988    83      52
1989   106      85
1990   120     114
;

proc computab data=example;

   columns c88 c89 c90 total;
   rows sales cgs gprofit pctmarg;

   /* calculate gross profit */
   gprofit = sales - cgs;

   /* select a column */
```

```
      c88 = year = 1988;
      c89 = year = 1989;
      c90 = year = 1990;

      /* calculate row totals for sales */
      /* and cost of goods sold */
      col: total = c88 + c89 + c90;

      /* calculate profit margin */
      row: pctmarg = gprofit / cgs * 100;
   run;
```

Table 13.2 shows the CDT before any observation is read in. All the columns and rows are defined with the values initialized to 0.

Table 13.2. CDT Before any Input

	C88	C89	C90	TOTAL
SALES	0	0	0	0
CGS	0	0	0	0
GPROFIT	0	0	0	0
PCTMARG	0	0	0	0

When the first input is read in (year=1988, sales=83, and cgs=52), the input block puts the values for SALES and CGS in the C88 column since year=1988. Also the value for the gross profit for that year (GPROFIT) is calculated as indicated in the following:

```
      gprofit = sales-cgs;
      c88 = year = 1988;
      c89 = year = 1989;
      c90 = year = 1990;
```

Table 13.3 shows the CDT after the first observation is input.

Table 13.3. CDT After First Observation Input (C88=1)

	C88	C89	C90	TOTAL
SALES	83	0	0	0
CGS	52	0	0	0
GPROFIT	31	0	0	0
PCTMARG	0	0	0	0

Similarly, the second observation (year=1989, sales=106, cgs=85) is put in the second column and the GPROFIT is calculated to be 21. The third observation (year=1990, sales=120, cgs=114) is put in the third column and the GPROFIT is calculated to be 6. Table 13.4 shows the CDT after all observations are input.

Table 13.4. CDT After All Observations Input

	C88	C89	C90	TOTAL
SALES	83	106	120	0
CGS	52	85	114	0
GPROFIT	31	21	6	0
PCTMARG	0	0	0	0

After the input block is executed for each observation in the input data set, the first row or column block is processed. In this case, the column block is

```
col: total = c88 + c89 + c90;
```

The column block executes for each row, calculating the TOTAL column for each row. Table 13.5 shows the CDT after the column block has executed for the first row (total=83 + 106 + 120). The total sales for the three years is 309.

Table 13.5. CDT After Column Block Executed for First Row

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	0
GPROFIT	31	21	6	0
PCTMARG	0	0	0	0

Table 13.6 shows the CDT after the column block has executed for all rows and the values for total cost of goods sold and total gross profit have been calculated.

Table 13.6. CDT After Column Block Executed for All Rows

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	251
GPROFIT	31	21	6	58
PCTMARG	0	0	0	0

Once the column block has been executed for all rows, the next block is processed. The row block is

```
row: pctmarg = gprofit / cgs * 100;
```

The row block executes for each column, calculating the PCTMARG for each year and the total (TOTAL column) for three years. Table 13.7 shows the CDT after the row block has executed for all columns.

Table 13.7. CDT After Row Block Executed for All Columns

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	251
GPROFIT	31	21	6	58
PCTMARG	59.62	24.71	5.26	23.11

Order of Calculations

The COMPUTAB procedure provides alternative programming methods for performing most calculations. New column and row values are formed by adding values from the input data set, directly or with modification, into existing columns or rows. New columns can be formed in the input block or in column blocks. New rows can be formed in the input block or in row blocks.

This example illustrates the different ways to collect totals. Table 13.8 is the total sales report for two products, SALES1 and SALES2, during the years 1988-1990. The values for SALES1 and SALES2 in columns C88, C89, and C90 come from the input data set.

Table 13.8. Total Sales Report

	C88	C89	C90	SALESTOT
SALES1	15	45	80	140
SALES2	30	40	50	120
YRTOT	45	85	130	260

The new column SALESTOT, which is the total sales for each product over three years, can be computed in several different ways:

- in the input block by selecting SALESTOT for each observation

```
salestot = 1;
```

- in a column block

```
coltot: salestot = c88 + c89 + c90;
```

In a similar fashion, the new row YRTOT, which is the total sales for each year, can be formed as follows:

- in the input block

```
yrtot = sales1 + sales2;
```

in a row block

```
rowtot: yrtot = sales1 + sales2;
```

Performing some calculations in PROC COMPUTAB in different orders can yield different results, since many operations are not commutative. Be sure to perform calculations in the proper sequence. It may take several column and row blocks to produce the desired report values.

Notice that in the previous example, the grand total for all rows and columns is 260 and is the same whether it is calculated from row subtotals or column subtotals. It makes no difference in this case whether you compute the row block or the column block first.

However, consider the following example where a new column and a new row are formed:

Table 13.9. Report Sensitive to Order of Calculations

	STORE1	STORE2	STORE3	MAX
PRODUCT1	12	13	27	27
PRODUCT2	11	15	14	15
TOTAL	23	28	41	?

The new column MAX contains the maximum value in each row, and the new row TOTAL contains the column totals. MAX is calculated in a column block:

```
   col: max = max(store1,store2,store3);
```

TOTAL is calculated in a row block:

```
   row: total = product1 + product2;
```

Notice that either of two values, 41 or 42, is possible for the element in column MAX and row TOTAL. If the row block is first, the value is the maximum of the column totals (41). If the column block is first, the value is the sum of the MAX values (42). Whether to compute a column block before a row block can be a critical decision.

Column Selection

The following discussion assumes that the NOTRANS option has not been specified. When NOTRANS is specified, this section applies to rows rather than columns.

If a COLUMNS statement appears in PROC COMPUTAB, a target column must be selected for the incoming observation. If there is no COLUMNS statement, a new column is added for each observation. When a COLUMNS statement is present and the selection criteria fail to designate a column, the current observation is ignored. Faulty column selection can result in columns or entire tables of 0s (or missing values if the INITMISS option is specified).

During execution of the input block, when an observation is read, its values are copied into row variables in the Program Data Vector (PDV).

To select columns, use either the column variable names themselves or the special variable _COL_. Use the column names by setting a column variable equal to some nonzero value. The example in the section "Getting Started" earlier in this chapter uses the logical expression COMPDIV= *value* which is evaluated to produce 0 or 1, and the result is assigned to the corresponding column variable.

```
a = compdiv = ’A’;
b = compdiv = ’B’;
c = compdiv = ’C’;
```

IF statements can also be used to select columns. The following statements are equivalent to the preceding example:

```
if      compdiv = ’A’ then a = 1;
else if compdiv = ’B’ then b = 1;
else if compdiv = ’C’ then c = 1;
```

At the end of the input block for each observation, PROC COMPUTAB multiplies numeric input values by any nonzero selector values and adds the result to selected columns. Character values simply overwrite the contents already in the table. If more than one column is selected, the values are added to each of the selected columns.

Use the _COL_ variable to select a column by assigning the column number to it. The COMPUTAB procedure automatically initializes column variables and sets the _COL_ variable to 0 at the start of each execution of the input block. At the end of the input block for each observation, PROC COMPUTAB examines the value of _COL_. If the value is nonzero and within range, the row variable values are added to the CDT cells of the _COL_th column, for example,

```
data rept;
   input div sales cgs;
   datalines;
2   106    85
3   120   114
1    83    52
;

proc computab data=rept;
   row div sales cgs;
   columns div1 div2 div3;
   _col_ = div;
run;
```

The code in this example places the first observation (DIV=2) in column 2 (DIV2), the second observation (DIV=3) in column 3 (DIV3), and the third observation (DIV=1) in column 1 (DIV1).

Controlling Execution within Row and Column Blocks

Row names, column names, and the special variables _ROW_ and _COL_ can be used to limit the execution of programming statements to selected rows or columns. A row block operates on all columns of the table for a specified row unless restricted in some way. Likewise, a column block operates on all rows for a specified column. Use column names or _COL_ in a row block to execute programming statements conditionally; use row names or _ROW_ in a column block.

For example, consider a simple column block consisting of only one statement:

```
col: total = qtr1 + qtr2 + qtr3 + qtr4;
```

This column block assigns a value to each row in the TOTAL column. As each row participates in the execution of a column block,

- its row variable in the program data vector is set to 1
- the value of _ROW_ is the number of the participating row
- the value from each column of the row is copied from the COMPUTAB data table to the program data vector

To avoid calculating TOTAL on particular rows, use row names or _ROW_, for example,

```
col: if sales|cost then total = qtr1 + qtr2 + qtr3 + qtr4;
```

or

```
col: if _row_ < 3  then total = qtr1 + qtr2 + qtr3 + qtr4;
```

Row and column blocks can appear in any order, and rows and columns can be selected in each block.

Program Flow

This section describes in detail the different steps in PROC COMPUTAB execution.

Step 1: Define Report Organization and Set Up the COMPUTAB Data Table

Before the COMPUTAB procedure reads in data or executes programming statements, the columns list from the COLUMNS statements and the rows list from the ROWS statements are used to set up a matrix of all columns and rows in the report. This matrix is called the COMPUTAB data table (CDT). When you define columns and rows of the CDT, the COMPUTAB procedure also sets up corresponding variables in working storage called the program data vector (PDV) for programming statements. Data values reside in the CDT but are copied into the program data vector as they are needed for calculations.

Step 2: Select Input Data with Input Block Programming Statements

The input block copies input observations into rows or columns of the CDT. By default, observations go to columns; if the data set is not transposed (NOTRANS option), observations go to rows of the report table. The input block consists of all executable statements before any ROW*xxxxx*: or COL*xxxxx*: statement label. Use programming statements to perform calculations and select a given observation to be added into the report.

Input Block

The input block is executed once for each observation in the input data set. If there is no input data set, the input block is not executed. The program logic of the input block is as follows:

1. Determine which variables, row or column, are selector variables and which are data variables. Selector variables determine which rows or columns receive values at the end of the block. Data variables contain the values that the selected rows or columns receive. By default, column variables are selector variables and row variables are data variables. If the input data set is not transposed (NOTRANS option), the roles are reversed.
2. Initialize nonretained program variables (including selector variables) to 0 (or missing if the INITMISS option is specified). Selector variables are temporarily associated with a numeric data item supplied by the procedure. Using these variables to control row and column selection does not affect any other data values.
3. Transfer data from an observation in the data set to data variables in the PDV.
4. Execute the programming statements in the input block using values from the PDV and storing results in the PDV.
5. Transfer data values from the PDV into the appropriate columns of the CDT. If a selector variable for a row or column has a nonmissing, nonzero value, multiply each PDV value for variables used in the report by the selector variable and add the results to the selected row or column of the CDT.

Step 3: Calculate Final Values Using Column Blocks and Row Blocks

Column Blocks

A column block is executed once for each row of the CDT. The program logic of a column block is as follows:

1. Indicate the current row by setting the corresponding row variable in the PDV to 1 and the other row variables to missing. Assign the current row number to the special variable _ROW_.
2. Move values from the current row of the CDT to the respective column variables in the PDV.
3. Execute programming statements in the column block using the column values in the PDV. Here, new columns can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current row of the CDT.

Row Blocks

A row block is executed once for each column of the CDT. The program logic of a row block is as follows:

1. Indicate the current column by setting the corresponding column variable in the PDV to 1 and the other column variables to missing. Assign the current column number to the special variable _COL_.
2. Move values from the current column of the CDT to the respective row variables in the PDV.
3. Execute programming statements in the row block using the row values in the PDV. Here new rows can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current column of the CDT.

See "Controlling Execution within Row and Column Blocks" later in this chapter for details.

Any number of column blocks and row blocks can be used. Each may include any number of programming statements.

The values of row variables and column variables are determined by the order in which different row-block and column-block programming statements are processed. These values can be modified throughout the COMPUTAB procedure, and final values are printed in the report.

Direct Access to Table Cells

You can insert or retrieve numeric values from specific table cells using the special reserved name TABLE with row and column subscripts. References to the TABLE have the form

```
TABLE[ row-index, column-index ]
```

where *row-index* and *column-index* can be numbers, character literals, numeric variables, character variables, or expressions that produce a number or a name. If an index is numeric, it must be within range; if it is character, it must name a row or column.

References to TABLE elements can appear on either side of an equal sign in an assignment statement and can be used in a SAS expression.

Reserved Words

Certain words are reserved for special use by the COMPUTAB procedure, and using these words as variable names can lead to syntax errors or warnings. They are:

- COLUMN
- COLUMNS
- COL
- COLS
- _COL_
- ROW
- ROWS
- _ROW_
- INIT
- _N_
- TABLE

Missing Values

Missing values for variables in programming statements are treated in the same way that missing values are treated in the DATA step; that is, missing values used in expressions propagate missing values to the result. Refer to *SAS Language: Reference* for more information about missing values.

Missing values in the input data are treated as follows in the COMPUTAB report table. At the end of the input block, either one or more rows or one or more columns may have been selected to receive values from the program data vector (PDV). Numeric data values from variables in the PDV are added into selected report table rows or columns. If a PDV value is missing, the values already in the selected rows or columns for that variable are unchanged by the current observation. Other values from the current observation are added to table values as usual.

OUT= Data Set

The output data set contains the following variables:

- BY variables
- a numeric variable _TYPE_
- a character variable _NAME_
- the column variables from the COMPUTAB data table

The BY variables contain values for the current BY group. For observations in the output data set from consolidation tables, the consolidated BY variables have missing values.

The special variable _TYPE_ is a numeric variable that can have one of three values: 1, 2, or 3. _TYPE_= 1 indicates observations from the normal report table produced for each BY group; _TYPE_= 2 indicates observations from the _TOTAL_ consolidation table; _TYPE_= 3 indicates observations from other consolidation tables. _TYPE_= 2 and 3 observations have one or more BY variables with missing values.

The special variable _NAME_ is a character variable of length 8 that contains the row or column name associated with the observation from the report table. If the input data set is transposed, _NAME_ contains column names; otherwise, _NAME_ contains row names.

If the input data set is transposed, the remaining variables in the output data set are row variables from the report table. They are column variables if the input data set is not transposed.

Examples

Example 13.1. Using Programming Statements

This example illustrates two ways of operating on the same input variables and producing the same tabular report. To simplify the example, no report enhancements are shown.

The manager of a hotel chain wants a report that shows the number of bookings at its hotels in each of four cities, the total number of bookings in the current quarter, and the percentage of the total coming from each location for each quarter of the year. Input observations contain the following variables: REPTDATE (report date), LA (number of bookings in Los Angeles), ATL (number of bookings in Atlanta), CH (number of bookings in Chicago), and NY (number of bookings in New York).

The following DATA step creates the SAS data set BOOKINGS:

```
data bookings;
   input reptdate date9. la atl ch ny;
   datalines;
01JAN1989 100 110 120 130
01FEB1989 140 150 160 170
01MAR1989 180 190 200 210
01APR1989 220 230 240 250
01MAY1989 260 270 280 290
01JUN1989 300 310 320 330
01JUL1989 340 350 360 370
01AUG1989 380 390 400 410
01SEP1989 420 430 440 450
01OCT1989 460 470 480 490
01NOV1989 500 510 520 530
01DEC1989 540 550 560 570
;
```

The following PROC COMPUTAB statements select columns by setting _COL_ to an appropriate value. The PCT1, PCT2, PCT3, and PCT4 columns represent the percentage contributed by each city to the total for the quarter. These statements produce Output 13.1.1.

```
proc computab data=bookings cspace=1 cwidth=6;

   columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
   columns qtr1-qtr4 / format=6.;
   columns pct1-pct4 / format=6.2;
   rows la atl ch ny total;

   /* column selection */
   _col_ = qtr( reptdate ) * 2 - 1;

   /* copy qtr column values temporarily into pct columns */
   colcopy:
      pct1 = qtr1;
      pct2 = qtr2;
      pct3 = qtr3;
      pct4 = qtr4;

   /* calculate total row for all columns */
   /* calculate percentages for all rows in pct columns only  */
   rowcalc:
      total = la + atl + ch + ny;
      if mod( _col_, 2 ) = 0 then do;
         la  = la  / total * 100;
         atl = atl / total * 100;
         ch  = ch  / total * 100;
         ny  = ny  / total * 100;
         total = 100;
         end;
run;
```

Output 13.1.1. Quarterly Report of Hotel Bookings

```
                 QTR1   PCT1   QTR2   PCT2   QTR3   PCT3   QTR4   PCT4


       LA         420  22.58    780  23.64   1140  24.05   1500  24.27
       ATL        450  24.19    810  24.55   1170  24.68   1530  24.76
       CH         480  25.81    840  25.45   1200  25.32   1560  25.24
       NY         510  27.42    870  26.36   1230  25.95   1590  25.73
       TOTAL     1860 100.00   3300 100.00   4740 100.00   6180 100.00
```

Using the same input data, the next set of statements shows the usefulness of arrays in allowing PROC COMPUTAB to work in two directions at once. Arrays in larger programs can both reduce the amount of program source code and simplify otherwise complex methods of referring to rows and columns. The same report as in Output 13.1.1 is produced.

```
proc computab data=bookings cspace=1 cwidth=6;
```

```
      columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
      columns qtr1-qtr4 / format=6.;
      columns pct1-pct4 / format=6.2;
      rows la atl ch ny total;

      array pct[4] pct1-pct4;
      array qt[4] qtr1-qtr4;
      array rowlist[5] la atl ch ny total;

      /* column selection */
      _col_ = qtr(reptdate) * 2 - 1;

      /* copy qtr column values temporarily into pct columns */
      colcopy:
         do i = 1 to 4;
            pct[i] = qt[i];
            end;

      /* calculate total row for all columns */
      /* calculate percentages for all rows in pct columns only */

      rowcalc:
         total = la + atl + ch + ny;
         if mod(_col_,2) = 0 then
            do i = 1 to 5;
               rowlist[i] = rowlist[i] / total * 100;
            end;
   run;
```

Example 13.2. Enhancing a Report

The following example shows how a report can be enhanced from a simple listing to a complex report. The simplest COMPUTAB report is a transposed listing of the data in the SAS data set INCOMREP shown in Output 13.2.1. To produce this output, nothing is specified except the PROC COMPUTAB statement and a TITLE statement.

```
data incomrep;
   input type :$8. date :monyy7.
         sales retdis tcos selling randd
         general admin deprec other taxes;
   format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 -8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14  0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15  2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 -8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15  2 490
;
title 'Computab Report without Any Specifications';
proc computab data=incomrep;
run;
```

Output 13.2.1. Simple Report

```
                    Computab Report without Any Specifications

                       COL1       COL2       COL3       COL4       COL5

         type        BUDGET     BUDGET     BUDGET     ACTUAL     ACTUAL
         date       JAN1989    FEB1989    MAR1989    JAN1989    FEB1989
         sales      4600.00    4700.00    4800.00    4900.00    5100.00
         retdis      300.00     330.00     360.00     505.00     480.00
         tcos       2200.00    2300.00    2600.00    2100.00    2400.00
         selling     480.00     500.00     500.00     430.00     510.00
         randd       110.00     110.00     120.00     130.00     110.00
         general     500.00     500.00     600.00     410.00     390.00
         admin       210.00     200.00     250.00     200.00     230.00
         deprec       14.00      14.00      15.00      14.00      15.00
         other        -8.00       0.00       2.00      -8.00       2.00
         taxes       510.00     480.00     520.00     500.00     490.00
```

To exclude the budgeted values from your report, select columns for ACTUAL observations only. To remove unwanted variables, specify the variables you want in a ROWS statement.

```
title 'Column Selection by Month';

proc computab data=incomrep;
   rows sales--other;
   columns jana feba mara;
   mnth = month(date);
   if type = 'ACTUAL';
      jana = mnth = 1;
      feba = mnth = 2;
      mara = mnth = 3;
run;
```

The report is shown in Output 13.2.2.

Output 13.2.2. Report Using Column Selection Techniques

```
                            Column Selection by Month

                                  JANA       FEBA       MARA

                    sales      4900.00    5100.00       0.00
                    retdis      505.00     480.00       0.00
                    tcos       2100.00    2400.00       0.00
                    selling     430.00     510.00       0.00
                    randd       130.00     110.00       0.00
                    general     410.00     390.00       0.00
                    admin       200.00     230.00       0.00
                    deprec       14.00      15.00       0.00
                    other        -8.00       2.00       0.00
```

To complete the report, compute new rows from existing rows. This is done in a row block (although it can also be done in the input block). Add a new column (QTR1) that accumulates all the actual data. The NOZERO option suppresses the zero column for March. The output produced by these statements is shown in Output 13.2.3.

```
proc computab data=incomrep;

   /* add a new column to be selected */
   /* qtr1 column will be selected several times */
   columns actual1-actual3 qtr1 / nozero;
   array collist[3] actual1-actual3;
   rows sales retdis netsales tcos grosspft selling randd general
        admin deprec operexp operinc other taxblinc taxes netincom;

   if type='ACTUAL';
   i = month(date);
   if i <= 3 then qtr1 = 1;
   collist[i]=1;

   rowcalc:
      if sales = . then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp  = selling + randd + general + admin + deprec;
      operinc  = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;
run;
```

Output 13.2.3. Report Using Techniques to Compute New Rows

```
                           Column Selection by Month

                           ACTUAL1    ACTUAL2       QTR1

                 SALES     4900.00    5100.00   10000.00
                 RETDIS     505.00     480.00     985.00
                 NETSALES  4395.00    4620.00    9015.00
                 TCOS      2100.00    2400.00    4500.00
                 GROSSPFT  2295.00    2220.00    4515.00
                 SELLING    430.00     510.00     940.00
                 RANDD      130.00     110.00     240.00
                 GENERAL    410.00     390.00     800.00
                 ADMIN      200.00     230.00     430.00
                 DEPREC      14.00      15.00      29.00
                 OPEREXP   1184.00    1255.00    2439.00
                 OPERINC   1111.00     965.00    2076.00
                 OTHER       -8.00       2.00      -6.00
                 TAXBLINC  1103.00     967.00    2070.00
                 TAXES      500.00     490.00     990.00
                 NETINCOM   603.00     477.00    1080.00
```

Now that you have all the numbers calculated, add specifications to improve the report's appearance. Specify titles, row and column labels, and formats. The report produced by these statements is shown in Output 13.2.4.

```
/* now get the report to look the way we want it */
title  'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Period to Date Actual';
title4 'Amounts in Thousands';

proc computab data=incomrep;

   columns actual1-actual3 qtr1 / nozero f=comma7. +3 ' ';
   array collist[3] actual1-actual3;
   columns actual1 / 'Jan';
   columns actual2 / 'Feb';
   columns actual3 / 'Mar';
   columns qtr1 / 'Total' 'Qtr 1';
   rows sales    / ' '
                   'Gross Sales ';
   rows retdis   / 'Less Returns & Discounts';
   rows netsales / 'Net Sales'                   +3 ol;
   rows tcos     / ' '
                   'Total Cost of Sales';
   rows grosspft / ' '
                   'Gross Profit';
   rows selling  / ' '
                   'Operating Expenses:'
                   '   Selling';
   rows randd    / '   R & D';
   rows general  / +3;
   rows admin    / '   Administrative';
   rows deprec   / '   Depreciation'             ul;
   rows operexp  / ' '                         skip;
   rows operinc  / 'Operating Income';
   rows other    / 'Other Income/-Expense'       ul;
   rows taxblinc / 'Taxable Income';
   rows taxes    / 'Income Taxes'                ul;
   rows netincom / '   Net Income'              dul;

   if type = 'ACTUAL';
   i = month( date );
   collist[i] = 1;

   colcalc:
      qtr1 = actual1 + actual2 + actual3;

   rowcalc:
      if sales = . then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp = selling + randd + general + admin + deprec;
      operinc = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;
run;
```

Output 13.2.4. Specifying Titles, Row and Column Labels, and Formats

```
                         Pro Forma Income Statement
                        XYZ Computer Services, Inc.
                           Period to Date Actual
                            Amounts in Thousands


                                                                 Total
                                        Jan           Feb        Qtr 1

          Gross Sales                 4,900         5,100       10,000
          Less Returns & Discounts      505           480          985
                                  ---------     ---------    ---------
             Net Sales                4,395         4,620        9,015

          Total Cost of Sales         2,100         2,400        4,500

          Gross Profit                2,295         2,220        4,515

          Operating Expenses:
             Selling                    430           510          940
             R & D                      130           110          240
             GENERAL                    410           390          800
             Administrative             200           230          430
             Depreciation                14            15           29
                                  ---------     ---------    ---------
                                      1,184         1,255        2,439

          Operating Income            1,111           965        2,076
          Other Income/-Expense          -8             2           -6
                                  ---------     ---------    ---------
          Taxable Income              1,103           967        2,070
          Income Taxes                  500           490          990
                                  ---------     ---------    ---------
             Net Income                 603           477        1,080
                                  =========     =========    =========
```

Example 13.3. Comparison of Actual and Budget

This example shows a more complex report that compares the actual data with the budgeted values. The same input data as in the previous example is used.

The report produced by these statements is shown in Output 13.3.1. The report shows the values for the current month and the year-to-date totals for budgeted amounts, actual amounts, and the actuals as a percentage of the budgeted amounts. The data have the values for January and February. Therefore, the CURMO variable (current month) in the RETAIN statement is set to 2. The values for the observations where the month of the year is 2 (February) are accumulated for the Current Month values. The year-to-date values are accumulated from those observations where the month of the year is less than or equal to 2 (January and February).

```
/* do a more complex report */
title  'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Budget Analysis';
title4 'Amounts in Thousands';
```

```
proc computab data=incomrep;

   columns cmbud cmact cmpct ytdbud ytdact ytdpct /
           zero=' ';
   columns cmbud--cmpct / mtitle='- Current Month: February -';
   columns ytdbud--ytdpct / mtitle='- Year To Date -';
   columns cmbud ytdbud / 'Budget' f=comma6.;
   columns cmact ytdact / 'Actual' f=comma6.;
   columns cmpct ytdpct / '%  ' f=7.2;
   columns cmbud--ytdpct / '-';
   columns ytdbud / _titles_;
   retain curmo 2; /* current month: February */
   rows sales    / ' '
                   'Gross Sales';
   rows retdis   / 'Less Returns & Discounts';
   rows netsales / 'Net Sales'                   +3 ol;
   rows tcos     / ' '
                   'Total Cost of Sales';
   rows grosspft / ' '
                   'Gross Profit'                +3;
   rows selling  / ' '
                   'Operating Expenses:'
                   '   Selling';
   rows randd    / '   R & D';
   rows general  / +3;
   rows admin    / '   Administrative';
   rows deprec   / '   Depreciation'             ul;
   rows operexp  / ' ';
   rows operinc  / 'Operating Income'           ol;
   rows other    / 'Other Income/-Expense'       ul;
   rows taxblinc / 'Taxable Income';
   rows taxes    / 'Income Taxes'                ul;
   rows netincom / '   Net Income'               dul;

   cmbud = type = 'BUDGET' & month(date) = curmo;
   cmact = type = 'ACTUAL' & month(date) = curmo;
   ytdbud = type = 'BUDGET' & month(date) <= curmo;
   ytdact = type = 'ACTUAL' & month(date) <= curmo;

   rowcalc:
      if cmpct | ytdpct then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp  = selling + randd + general + admin + deprec;
      operinc  = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;

   colpct:
      if cmbud  & cmact  then cmpct  = 100 * cmact  / cmbud;
      if ytdbud & ytdact then ytdpct = 100 * ytdact / ytdbud;
run;
```

Output 13.3.1. Report Using Specifications to Tailor Output

```
                                 Pro Forma Income Statement
                                XYZ Computer Services, Inc.
                                      Budget Analysis
                                    Amounts in Thousands

   --- Current Month: February ---                            -------- Year To Date ---------
      Budget      Actual        %                                Budget      Actual        %
   ---------   ---------   ---------                          ---------   ---------   ---------

       4,700       5,100     108.51  Gross Sales                  9,300      10,000      107.53
         330         480     145.45  Less Returns & Discounts       630         985      156.35
   ---------   ---------   ---------                          ---------   ---------   ---------
       4,370       4,620     105.72     Net Sales                 8,670       9,015      103.98

       2,300       2,400     104.35  Total Cost of Sales          4,500       4,500      100.00

       2,070       2,220     107.25     Gross Profit              4,170       4,515      108.27

                                     Operating Expenses:
         500         510     102.00     Selling                     980         940       95.92
         110         110     100.00     R & D                       220         240      109.09
         500         390      78.00     GENERAL                   1,000         800       80.00
         200         230     115.00     Administrative              410         430      104.88
          14          15     107.14     Depreciation                 28          29      103.57
   ---------   ---------   ---------                          ---------   ---------   ---------
       1,324       1,255      94.79                               2,638       2,439       92.46
   ---------   ---------   ---------                          ---------   ---------   ---------
         746         965     129.36  Operating Income             1,532       2,076      135.51
                       2             Other Income/-Expense           -8          -6       75.00
   ---------   ---------   ---------                          ---------   ---------   ---------
         746         967     129.62  Taxable Income               1,524       2,070      135.83
         480         490     102.08  Income Taxes                   990         990      100.00
   ---------   ---------   ---------                          ---------   ---------   ---------
         266         477     179.32     Net Income                  534       1,080      202.25
   =========   =========   =========                          =========   =========   =========
```

Example 13.4. Consolidations

This example consolidates product tables by region and region tables by corporate division. Output 13.4.1 shows the North Central and Northeast regional summaries for the Equipment division for the first quarter. Output 13.4.2 shows the profit summary for the Equipment division. Similar tables for the Publishing division are produced but not shown here.

```
data product;
   input pcode div region month sold revenue recd cost;
datalines;
1 1 1 1 56 5600 29 2465
1 1 1 2 13 1300 30 2550
1 1 1 3 17 1700 65 5525
2 1 1 1  2  240 50 4900
2 1 1 2 82 9840 17 1666
more data lines
;

proc format;
   value divfmt 1='Equipment'
                2='Publishing';
   value regfmt 1='North Central'
                2='Northeast'
                3='South'
                4='West';
```

```
run;

proc sort data=product;
   by div region pcode;
run;

title1 '     XYZ Development Corporation      ';
title2 ' Corporate Headquarters: New York, NY ';
title3 '           Profit Summary             ';
title4 '                                      ';


proc computab data=product sumonly;
   by div region pcode;
   sumby _total_ div region;

   format div    divfmt.;
   format region regfmt.;
   label  div = 'DIVISION';

   /* specify order of columns and column titles */
   columns jan feb mar qtr1 / mtitle='- first quarter -' ' '  nozero;
   columns apr may jun qtr2 / mtitle='- second quarter -' ' ' nozero;
   columns jul aug sep qtr3 / mtitle='- third quarter -' ' '  nozero;
   columns oct nov dec qtr4 / mtitle='- fourth quarter -' ' ' nozero;
   column  jan  / ' ' 'January' '=';
   column  feb  / ' ' 'February' '=';
   column  mar  / ' ' 'March' '=';
   column  qtr1 / 'Quarter' 'Summary' '=';

   column  apr  / ' ' 'April' '=' _page_;
   column  may  / ' ' 'May' '=';
   column  jun  / ' ' 'June' '=';
   column  qtr2 / 'Quarter' 'Summary' '=';

   column  jul  / ' ' 'July' '=' _page_;
   column  aug  / ' ' 'August' '=';
   column  sep  / ' ' 'September' '=';
   column  qtr3 / 'Quarter' 'Summary' '=';

   column  oct  / ' ' 'October' '=' _page_;
   column  nov  / ' ' 'November' '=';
   column  dec  / ' ' 'December' '=';
   column  qtr4 / 'Quarter' 'Summary' '=';

   /* specify order of rows and row titles */
   row     sold    / ' ' 'Number Sold' f=8.;
   row     revenue / ' ' 'Sales Revenue';
   row     recd    / ' ' 'Number Received' f=8.;
   row     cost    / ' ' 'Cost of' 'Items Received';
   row     profit  / ' ' 'Profit' 'Within Period' ol;
   row     pctmarg / ' ' 'Profit Margin' dul;

   /* select column for appropriate month */
   _col_ = month + ceil( month / 3 ) - 1;

   /* calculate quarterly summary columns */
   colcalc:
      qtr1 = jan + feb + mar;
```

```
            qtr2 = apr + may + jun;
            qtr3 = jul + aug + sep;
            qtr4 = oct + nov + dec;

         /* calculate profit rows */
          rowcalc:
             profit = revenue - cost;
             if cost > 0 then pctmarg = profit / cost * 100;
      run;
```

Output 13.4.1. Summary by Regions for the Equipment Division

```
                          XYZ Development Corporation
                    Corporate Headquarters: New York, NY
                               Profit Summary


------------SUMMARY TABLE:  DIVISION=Equipment region=North Central------------

                            ------------- first quarter --------------

                                                                Quarter
                              January   February      March     Summary
                            =========  =========  =========  =========

          Number Sold             198        223        119        540

          Sales Revenue      22090.00   26830.00   14020.00   62940.00

          Number Received         255        217        210        682

          Cost of
          Items Received     24368.00   20104.00   19405.00   63877.00
                            ---------  ---------  ---------  ---------

          Profit
          Within Period      -2278.00    6726.00   -5385.00    -937.00

          Profit Margin         -9.35      33.46     -27.75      -1.47
                            =========  =========  =========  =========
                      XYZ Development Corporation
                  Corporate Headquarters: New York, NY
                             Profit Summary


--------------SUMMARY TABLE:  DIVISION=Equipment region=Northeast--------------

                           ------------- first quarter --------------

                                                                Quarter
                             January   February      March    Summary
                           =========  =========  =========  =========

          Number Sold             82        180        183        445

          Sales Revenue      9860.00   21330.00   21060.00   52250.00

          Number Received        162         67        124        353

          Cost of
          Items Received    16374.00    6325.00   12333.00   35032.00
                           ---------  ---------  ---------  ---------

          Profit
          Within Period     -6514.00   15005.00    8727.00   17218.00

          Profit Margin       -39.78     237.23      70.76      49.15
                           =========  =========  =========  =========
```

Output 13.4.2. Profit Summary for the Equipment Division

```
                      XYZ Development Corporation
                  Corporate Headquarters: New York, NY
                             Profit Summary


-----------------------SUMMARY TABLE:  DIVISION=Equipment----------------------

                           ------------- first quarter --------------

                                                                Quarter
                             January   February      March    Summary
                           =========  =========  =========  =========

          Number Sold            280        403        302        985

          Sales Revenue     31950.00   48160.00   35080.00  115190.00

          Number Received        417        284        334       1035

          Cost of
          Items Received    40742.00   26429.00   31738.00   98909.00
                           ---------  ---------  ---------  ---------

          Profit
          Within Period     -8792.00   21731.00    3342.00   16281.00

          Profit Margin       -21.58      82.22      10.53      16.46
                           =========  =========  =========  =========
```

Output 13.4.3 shows the consolidation report of profit summary over both divisions and regions.

Output 13.4.3. Profit Summary

```
                              XYZ Development Corporation
                          Corporate Headquarters: New York, NY
                                    Profit Summary


----------------------------SUMMARY TABLE:  TOTALS----------------------------

                           ------------- first quarter --------------

                                                                Quarter
                             January   February      March      Summary
                           =========  =========  =========  =========

          Number Sold            590        683        627       1900

          Sales Revenue     41790.00   55910.00   44800.00  142500.00

          Number Received        656        673        734       2063

          Cost of
          Items Received    46360.00   35359.00   40124.00  121843.00
                           ---------  ---------  ---------  ---------

          Profit
          Within Period     -4570.00   20551.00    4676.00   20657.00

          Profit Margin        -9.86      58.12      11.65      16.95
                           =========  =========  =========  =========
```

Example 13.5. Creating an Output Data Set

This example uses data and reports similar to those in Example 13.3 to illustrate the creation of an output data set.

```
data product;
   input pcode div region month sold revenue recd cost;
   datalines;
1 1 1 1 56 5600 29 2465
1 1 1 2 13 1300 30 2550
1 1 1 3 17 1700 65 5525
2 1 1 1  2  240 50 4900
2 1 1 2 82 9840 17 1666
more data lines
;

proc sort data=product out=sorted;
   by div region;
run;

/* create data set, profit */
proc computab data=sorted notrans out=profit noprint;
```

```
      by div region;
      sumby div;

      /* specify order of rows and row titles */
      row     jan feb mar qtr1;
      row     apr may jun qtr2;
      row     jul aug sep qtr3;
      row     oct nov dec qtr4;

      /* specify order of columns and column titles */
      columns sold revenue recd cost profit pctmarg;

      /* select row for appropriate month */
      _row_ = month + ceil( month / 3 ) - 1;

      /* calculate quarterly summary rows */
      rowcalc:
         qtr1 = jan + feb + mar;
         qtr2 = apr + may + jun;
         qtr3 = jul + aug + sep;
         qtr4 = oct + nov + dec;

      /* calculate profit columns */
      colcalc:
         profit = revenue - cost;
         if cost > 0 then pctmarg = profit / cost * 100;
   run;

      /* make a partial listing of the output data set */
   proc print data=profit (obs=10) noobs;
   run;
```

Since the NOTRANS option is specified, column names become variables in the data set. REGION has missing values in the output data set for observations associated with consolidation tables. The output data set PROFIT, in conjunction with the option NOPRINT, illustrates how you can use the computational features of PROC COMPUTAB for creating additional rows and columns as in a spread sheet without producing a report. Output 13.5.1 shows a partial listing of the output data set PROFIT.

Output 13.5.1. Partial Listing of the PROFIT Data Set

```
div  region  _TYPE_  _NAME_  sold  revenue  recd   cost  PROFIT   PCTMARG

 1      1       1     JAN     198   22090    255  24368   -2278    -9.348
 1      1       1     FEB     223   26830    217  20104    6726    33.456
 1      1       1     MAR     119   14020    210  19405   -5385   -27.751
 1      1       1     QTR1    540   62940    682  63877    -937    -1.467
 1      1       1     APR      82    9860    162  16374   -6514   -39.783
 1      1       1     MAY     180   21330     67   6325   15005   237.233
 1      1       1     JUN     183   21060    124  12333    8727    70.761
 1      1       1     QTR2    445   52250    353  35032   17218    49.149
 1      1       1     JUL     194   23210     99  10310   12900   125.121
 1      1       1     AUG     153   17890    164  16704    1186     7.100
```

Example 13.6. A What-If Market Analysis

PROC COMPUTAB can be used with other SAS/ETS procedures and with macros to implement commonly needed decision support tools for financial and marketing analysis.

The following input data set reads quarterly sales figures:

```
data market;
   input date :yyq6. units @@;
   datalines;
1980Q1  3608.9  1980Q2  5638.4  1980Q3  6017.9  1980Q4  4929.6
1981Q1  4962.0  1981Q2  5804.6  1981Q3  5498.6  1981Q4  7687.1
1982Q1  6864.1  1982Q2  7625.8  1982Q3  7919.7  1982Q4  8294.7
1983Q1  8151.6  1983Q2 10992.7  1983Q3 10671.4  1983Q4 10643.2
1984Q1 10215.1  1984Q2 10795.5  1984Q3 14144.4  1984Q4 11623.1
1985Q1 14445.3  1985Q2 13925.2  1985Q3 16729.3  1985Q4 16125.3
1986Q1 15232.6  1986Q2 16272.2  1986Q3 16816.7  1986Q4 17040.0
1987Q1 17967.8  1987Q2 14727.2  1987Q3 18797.3  1987Q4 18258.0
1988Q1 20041.5  1988Q2 20181.0  1988Q3 20061.7  1988Q4 21670.1
1989Q1 21844.3  1989Q2 23524.1  1989Q3 22000.6  1989Q4 24166.7
;
```

PROC FORECAST makes a total market forecast for the next four quarters.

```
/* forecast the total number of units to be */
/* sold in the next four quarters */
proc forecast out=outcome trend=2 interval=qtr lead=4;
   id date;
   var units;
run;
```

The macros WHATIF and SHOW build a report table and provide the flexibility of examining alternate what-if situations. The row and column calculations of PROC COMPUTAB compute the income statement. With macros stored in a macro library, the only statements required with PROC COMPUTAB are macro invocations and TITLE statements.

```
/* set up rows and columns of report and initialize */
/* market share and program constants */
%macro whatif(mktshr=,price=,ucost=,taxrate=,numshar=,overhead=);

   columns mar / ' ' 'March';
   columns jun / ' ' 'June';
   columns sep / ' ' 'September';
   columns dec / ' ' 'December';
   columns total / 'Calculated' 'Total';
   rows mktshr / 'Market Share'          f=5.2;
   rows tunits / 'Market Forecast';
   rows units  / 'Items Sold';
   rows sales  / 'Sales';
   rows cost   / 'Cost of Goods';
   rows ovhd   / 'Overhead';
   rows gprof  / 'Gross Profit';
   rows tax    / 'Tax';
   rows pat    / 'Profit After Tax';
   rows earn   / 'Earnings per Share';

   rows mktshr--earn / skip;
   rows sales--earn  /  f=dollar12.2;
   rows tunits units /  f=comma12.2;

   /* initialize market share values */
   init mktshr &mktshr;

   /* define constants */
   retain price &price ucost &ucost taxrate &taxrate
          numshar &numshar;

   /* retain overhead and sales from previous quarter */
   retain prevovhd &overhead prevsale;
   %mend whatif;


/* perform calculations and print the specified rows */
%macro show(rows);

   /* initialize list of row names */
   %let row1  = mktshr;
   %let row2  = tunits;
   %let row3  = units;
   %let row4  = sales;
   %let row5  = cost;
   %let row6  = ovhd;
   %let row7  = gprof;
   %let row8  = tax;
   %let row9  = pat;
   %let row10 = earn;

   /* find parameter row names in list and eliminate */
   /* them from the list of noprint rows */
   %let n = 1;
   %let word = %scan(&rows,&n);
   %do %while(&word NE );
      %let i = 1;
      %let row11 = &word;
      %do %while(&&row&i NE &word);
```

```
            %let i = %eval(&i+1);
            %end;
         %if &i<11 %then %let row&i = ;
         %let n = %eval(&n+1);
         %let word = %scan(&rows,&n);
      %end;

      rows &row1 &row2 &row3 &row4 &row5 &row6 &row7
           &row8 &row9 &row10 dummy / noprint;

      /* select column using lead values from proc forecast */
      mar = _lead_ = 1;
      jun = _lead_ = 2;
      sep = _lead_ = 3;
      dec = _lead_ = 4;

      rowreln:;
         /* inter-relationships */
         share  = round( mktshr, 0.01 );
         tunits = units;
         units  = share * tunits;
         sales  = units * price;
         cost   = units * ucost;

         /* calculate overhead */
         if mar then prevsale = sales;
         if sales > prevsale
            then ovhd = prevovhd + .05 * ( sales - prevsale );
            else ovhd = prevovhd;
         prevovhd = ovhd;
         prevsale = sales;
         gprof = sales - cost - ovhd;
         tax  = gprof * taxrate;
         pat  = gprof - tax;
         earn = pat / numshar;

      coltot:;
         if mktshr
            then total = ( mar + jun + sep + dec ) / 4;
            else total = mar + jun + sep + dec;
      %mend show;
      run;
```

The following PROC COMPUTAB statements use the PROC FORECAST output data set with invocations of the macros defined previously to perform a what-if analysis of the predicted income statement. The report is shown in Output 13.6.1.

```
   title1 'Fleet Footwear, Inc.';
   title2 'Marketing Analysis Income Statement';
   title3 'Based on Forecasted Unit Sales';
   title4 'All Values Shown';

   proc computab data=outcome cwidth=12;

      %whatif(mktshr=.02 .07 .15 .25,price=38.00,
              ucost=20.00,taxrate=.48,numshar=15000,overhead=5000);
      %show(mktshr tunits units sales cost ovhd gprof tax pat earn);
   run;
```

Output 13.6.1. PROC COMPUTAB Report Using Macro Invocations

```
                            Fleet Footwear, Inc.
                     Marketing Analysis Income Statement
                        Based on Forecasted Unit Sales
                               All Values Shown

                                                                        Calculated
                           March         June    September    December       Total
Market Share                0.02         0.07         0.15         0.25         0.12

Market Forecast        23,663.94    24,169.61    24,675.27    25,180.93    97,689.75

Items Sold                473.28     1,691.87     3,701.29     6,295.23    12,161.67

Sales                $17,984.60   $64,291.15  $140,649.03  $239,218.83  $462,143.61

Cost of Goods         $9,465.58   $33,837.45   $74,025.80  $125,904.65  $243,233.48

Overhead              $5,000.00    $7,315.33   $11,133.22   $16,061.71   $39,510.26

Gross Profit          $3,519.02   $23,138.38   $55,490.00   $97,252.47  $179,399.87

Tax                   $1,689.13   $11,106.42   $26,635.20   $46,681.19   $86,111.94

Profit After Tax      $1,829.89   $12,031.96   $28,854.80   $50,571.28   $93,287.93

Earnings per Share        $0.12        $0.80        $1.92        $3.37        $6.22
```

The following statements produce a similar report for different values of market share and unit costs. The report in Output 13.6.2 displays the values for the market share, market forecast, sales, after tax profit, and earnings per share.

```
   title3 'Revised';
   title4 'Selected Values Shown';

   proc computab data=outcome cwidth=12;
      %whatif(mktshr=.01 .06 .12 .20,price=38.00,
              ucost=23.00,taxrate=.48,numshar=15000,overhead=5000);
      %show(mktshr tunits sales pat earn);
   run;
```

Output 13.6.2. Report Using Macro Invocations for Selected Values

```
                                Fleet Footwear, Inc.
                          Marketing Analysis Income Statement
                                       Revised
                                 Selected Values Shown

                                                                          Calculated
                            March         June    September     December        Total
   Market Share              0.01         0.06         0.12         0.20         0.10

   Market Forecast      23,663.94    24,169.61    24,675.27    25,180.93    97,689.75

   Sales                $8,992.30   $55,106.70  $112,519.22  $191,375.06  $367,993.28

   Profit After Tax      $-754.21    $7,512.40   $17,804.35   $31,940.30   $56,502.84

   Earnings per Share     $-0.05        $0.50        $1.19        $2.13        $3.77
```

Example 13.7. Cash Flows

The COMPUTAB procedure can be used to model cash flows from one time period to the next. The RETAIN statement is useful for enabling a row or column to contribute one of its values to its successor. Financial functions such as IRR (internal rate of return) and NPV (net present value) can be used on PROC COMPUTAB table values to provide a more comprehensive report. The following statements produce Output 13.7.1:

```
data cashflow;
   input date date9. netinc depr borrow invest tax div adv ;
   datalines;
30MAR1982 65 42 32 126 43 51 41
30JUN1982 68 47 32 144 45 54 46
30SEP1982 70 49 30 148 46 55 47
30DEC1982 73 49 30 148 48 55 47
;

title1 'Blue Sky Endeavors';
title2 'Financial Summary';
title4 '(Dollar Figures in Thousands)';

proc computab data=cashflow;

   cols qtr1 qtr2 qtr3 qtr4 / 'Quarter' f=7.1;
   col  qtr1 / 'One';
   col  qtr2 / 'Two';
   col  qtr3 / 'Three';
   col  qtr4 / 'Four';
   row  begcash / 'Beginning Cash';
   row  netinc  / 'Income' '   Net income';
   row  depr    / 'Depreciation';
   row  borrow;
   row  subtot1 / 'Subtotal';
   row  invest  / 'Expenditures' '   Investment';
   row  tax     / 'Taxes';
   row  div     / 'Dividend';
   row  adv     / 'Advertising';
   row  subtot2 / 'Subtotal';
```

```
      row  cashflow/  skip;
      row  irret   / 'Internal Rate' 'of Return' zero=' ';
      rows depr borrow subtot1 tax div adv subtot2 / +3;

      retain cashin -5;
      _col_ = qtr( date );

      rowblock:
         subtot1 = netinc + depr + borrow;
         subtot2 = tax + div + adv;
         begcash = cashin;
         cashflow = begcash + subtot1 - subtot2;
         irret = cashflow;
         cashin = cashflow;

      colblock:
         if begcash then cashin = qtr1;
         if irret then do;
            temp = irr( 4, cashin, qtr1, qtr2, qtr3, qtr4 );
            qtr1 = temp;
            qtr2 = 0; qtr3 = 0; qtr4 = 0;
            end;
   run;
```

Output 13.7.1. Report Using a RETAIN Statement and the IRR Financial Function

```
                              Blue Sky Endeavors
                              Financial Summary

                        (Dollar Figures in Thousands)

                          Quarter    Quarter    Quarter    Quarter
                              One        Two      Three       Four
          Beginning Cash     -5.0       -1.0        1.0        2.0
          Income
             Net income      65.0       68.0       70.0       73.0
             Depreciation    42.0       47.0       49.0       49.0
             BORROW          32.0       32.0       30.0       30.0
             Subtotal       139.0      147.0      149.0      152.0
          Expenditures
             Investment     126.0      144.0      148.0      148.0
             Taxes           43.0       45.0       46.0       48.0
             Dividend        51.0       54.0       55.0       55.0
             Advertising     41.0       46.0       47.0       47.0
             Subtotal       135.0      145.0      148.0      150.0
          CASHFLOW           -1.0        1.0        2.0        4.0

          Internal Rate
          of Return          20.9
```

Chapter 14

The DATASOURCE Procedure

Chapter Contents

Chapter 14
The DATASOURCE Procedure

Overview

The DATASOURCE procedure extracts time series data from many different kinds of data files distributed by various data vendors and stores them in a SAS data set. Once stored in a SAS data set, the time series variables can be processed by other SAS procedures.

The DATASOURCE procedure has statements and options to extract only a subset of time series data from an input data file. It gives you control over the frequency of data to be extracted, time series variables to be selected, cross sections to be included, and the time range of data to be output.

The DATASOURCE procedure can create auxiliary data sets containing descriptive information on the time series variables and cross sections. More specifically, the OUTCONT= data set contains information on time series variables, the OUTBY= data set reports information on the cross-sectional variables, and the OUTALL= data set combines both time series variables and cross-sectional information.

The output variables in the output and auxiliary data sets can be assigned various attributes by the DATASOURCE procedure. These attributes are labels, formats, new names, and lengths. While the first three attributes in this list are used to enhance the output, the length attribute is used to control the memory and disk-space usage of the DATASOURCE procedure.

Data files currently supported by the DATASOURCE procedure include

- U.S. Bureau of Economic Analysis data files:
  - National Income and Product Accounts tapes
  - National Income and Product Accounts diskettes
  - S-page diskettes
- U.S. Bureau of Labor Statistics data files:
  - Consumer Price Index Surveys
  - Producer Price Index Survey
  - National Employment, Hours, and Earnings Survey
  - State and Area Employment, Hours, and Earnings Survey
- Standard & Poor’s Compustat Services Financial Database Files:
  - COMPUSTAT Annual
  - COMPUSTAT 48 Quarter

- Center for Research in Security Prices (CRSP) data files:
  - Daily Binary Format Files
  - Monthly Binary Format Files
  - Daily Character Format Files
  - Monthly Character Format Files
  - Daily IBM Binary Format Files
  - Monthly IBM Binary Format Files
  - 1995 CDROM Character Format Files
  - 1995 CDROM UNIX (SUN) Binary Format Files
  - ACCESS97 CDROM Binary Format Files
- DRI/McGraw-Hill data files:
  - Basic Economics Data (formerly CITIBASE)
  - DRI Data Delivery Service files
  - Tape Format CITIBASE Data Files
  - DRI Data Delivery Service Time Series
  - PC Diskette format CITIBASE Databases
- FAME Information Services Databases
- Haver Analytics data files
- International Monetary Fund's Economic Information System data files:
  - International Financial Statistics
  - Direction of Trade Statistics
  - Balance of Payment Statistics
  - Government Finance Statistics
- Organization for Economic Cooperation and Development:
  - Annual National Accounts
  - Quarterly National Accounts
  - Main Economic Indicators

Getting Started

Structure of a SAS Data Set Containing Time Series Data

SAS procedures require time series data to be in a specific form recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods.

The time periods at which observations are recorded can be included in the data set as a time ID variable. The DATASOURCE procedure does include a time ID variable by the name of DATE.

For example, the following data set, extracted from a CITIBASE data file, gives the foreign exchange rates for Japan, Switzerland, and the United Kingdom, respectively.

Time ID variable	Time Series Variables		
DATE	EXRJAN	EXRSW	EXRUK
SEP1987	143.290	1.50290	164.460
OCT1987	143.320	1.49400	166.200
NOV1987	135.400	1.38250	177.540
DEC1987	128.240	1.33040	182.880
JAN1988	127.690	1.34660	180.090
FEB1988	129.170	1.39160	175.820

Figure 14.1. The Form of SAS Data Sets Required by Most SAS/ETS Procedures

Reading Data Files

The DATASOURCE procedure is designed to read data from many different files and to place them in a SAS data set. For example, if you have a DRI Basic Economics data file you want to read, use the following statements:

```
proc datasource filetype=dribasic infile=citifile out=dataset;
run;
```

Here, the FILETYPE= option indicates that you want to read DRI's Basic Economics data file, the INFILE= option specifies the fileref CITIFILE of the external file you want to read, and the OUT= option names the SAS data set to contain the time series data.

Subsetting Input Data Files

When only a subset of a data file is needed, it is inefficient to extract all the data and then subset it in a subsequent DATA step. Instead, you can use the DATASOURCE procedure options and statements to extract only needed information from the data file.

The DATASOURCE procedure offers the following subsetting capabilities:

- the INTERVAL= option controls the frequency of data output
- the KEEP or DROP statements selects a subset of time series variables
- the RANGE statement restricts the time range of data
- the WHERE statement selects a subset of cross sections

Controlling the Frequency of Data – The INTERVAL= Option

The OUT= data set can only contain data with the same frequency. If the data file you want to read contains time series data with several frequencies, you can indicate the frequency of data you want to extract with the INTERVAL= option. For example, the following statements extract all monthly time series from the DRIBASIC file CITIFILE:

```
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
run;
```

When the INTERVAL= option is not given, the default frequency defined for the FILETYPE= type file is used. For example, the statements in the previous section extract yearly series since INTERVAL=YEAR is the default frequency for DRI's Basic Economic Data files.

To extract data for several frequencies, you need to execute the DATASOURCE procedure once for each frequency.

Selecting Time Series Variables – The KEEP and DROP Statements

If you want to include specific series in the OUT= data set, list them in a KEEP statement. If, on the other hand, you want to exclude some variables from the OUT= data set, list them in a DROP statement. For example, the following statements extract monthly foreign exchange rates for Japan (EXRJAN), Switzerland (EXRSW), and the United Kingdom (EXRUK) from a DRIBASIC file CITIFILE:

```
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
   keep  exrjan exrsw exruk;
run;
```

The KEEP statement also allows input names to be quoted strings. If the name of a series in the input file contains blanks or special characters that are not valid SAS name syntax, put the series name in quotes to select it. Another way to allow the use of special characters in your SAS variable names, is to use the SAS options statement to designate VALIDVARNAME= ANY. This option will allow PROC DATASOURCE to include special characters in your SAS variable names. The following is an example of extracting series from a FAME database using the DATASOURCE procedure.

```
proc datasource filetype=fame dbname='fame_nys /disk1/prc/prc'
                interval=weekday out=outds outcont=attrds;
   range '1jan90'd to '1feb90'd;
   keep cci.close
        '{ibm.high,ibm.low,ibm.close}'
        'mave(ibm.close,30)'
        'crosslist({gm,f,c},{volume})'
        'cci.close+ibm.close';
   rename 'mave(ibm.close,30)' = ibm30day
          'cci.close+ibm.close' = cci_ibm;
run;
```

The resulting output data set OUTDS contains the following series: DATE, CCI_CLOS, IBM_HIGH, IBM_LOW, IBM_CLOS, IBM30DAY, GM_VOLUM, F_VOLUME, C_VOLUME, CCI_IBM.

Obviously, to be able to use KEEP and DROP statements, you need to know the name of time series variables available in the data file. The OUTCONT= option gives you this information. More specifically, the OUTCONT= option creates a data set containing descriptive information on the same frequency time series. This descriptive information includes series names, a flag indicating if the series is selected for output, series variable types, lengths, position of series in the OUT= data set, labels, format names, format lengths, format decimals, and a set of FILETYPE= specific descriptor variables. For example, the following statements list some of the monthly series available in the CITIFILE:

```
filename citifile 'host-specific-file-name' <host-options>;
proc datasource filetype=dribasic infile=citifile
                interval=month  outcont=vars;
run;

title1 'Some Time Series Variables Available in CITIFILE';
proc print data=vars noobs;
run;
```

```
                Some Time Series Variables Available in CITIFILE

          s
          e
          l         l    v
          e         e    a                        l
  n       c    t    n    r                        a
  a       t    y    g    n                        b
  m       e    p    t    u                        e
  e       d    e    h    m                        l

EXRJAN    1    1    5    .    FOREIGN EXCHANGE RATE: JAPAN (YEN PER U.
EXRSW     1    1    5    .    FOREIGN EXCHANGE RATE: SWITZERLAND (SWIS
EXRUK     1    1    5    .    FOREIGN EXCHANGE RATE: UNITED KINGDOM (C

                            d
                            e                                    f   f
                            s                                f   o   o
                            c                                o   r   r
                            r                                r   m   m     c
                            i                                m   a   a     o
                            p                                a   t   t     d
                            t                                t   l   d     e

FOREIGN EXCHANGE RATE: JAPAN (YEN PER U.S.$)                     0   0   EXRJAN
FOREIGN EXCHANGE RATE: SWITZERLAND (SWISS FRANC PER U.S.$)       0   0   EXRSW
FOREIGN EXCHANGE RATE: UNITED KINGDOM (CENTS PER POUND)          0   0   EXRUK
```

Figure 14.2. Partial Listing of the OUTCONT= Data Set

Controlling the Time Range of Data – The RANGE Statement

The RANGE statement is used to control the time range of observations included in the output data set. For example, if you want to extract the foreign exchange rates from September, 1987 to February, 1988, you can use the following statements:

```
filename citifile 'host-specific-file-name' <host-options>;
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
   keep  exrjan exrsw exruk;
   range from 1987:9 to 1988:2;
run;

title1 'Printout of the OUT= Data Set';
proc print data=dataset noobs;
run;
```

```
                    Printout of the OUT= Data Set

                  date      exrjan      exrsw       exruk

                SEP1987     143.290     1.50290     164.460
                OCT1987     143.320     1.49400     166.200
                NOV1987     135.400     1.38250     177.540
                DEC1987     128.240     1.33040     182.880
                JAN1988     127.690     1.34660     180.090
                FEB1988     129.170     1.39160     175.820
```

Figure 14.3. Subset Obtained by KEEP and RANGE Statements

Reading in Data Files Containing Cross Sections

Some data files group time series data with respect to cross-section identifiers; for example, International Financial Statistics files, distributed by IMF, group data with respect to countries (COUNTRY). Within each country, data are further grouped by Control Source Code (CSC), Partner Country Code (PARTNER), and Version Code (VERSION).

If a data file contains cross-section identifiers, the DATASOURCE procedure adds them to the output data set as BY variables. For example, the data set in Table 14.1 contains three cross sections:

- the first one is identified by (COUNTRY='112' CSC='F' PARTNER=' ' VERSION='Z')
- the second one is identified by (COUNTRY='146' CSC='F' PARTNER=' ' VERSION='Z')
- the third one is identified by (COUNTRY='158' CSC='F' PARTNER=' ' VERSION='Z').

Table 14.1. The Form of a SAS Data Set Containing BY Variables

BY Variables				Time ID Variable	Time Series Variables	
COUNTRY	CSC	PARTNER	VERSION	DATE	EFFEXR	EXRINDEX
112	F		Z	SEP1987	9326	12685
112	F		Z	OCT1987	9393	12813
112	F		Z	NOV1987	9626	13694
112	F		Z	DEC1987	9675	14099
112	F		Z	JAN1988	9581	13910
112	F		Z	FEB1988	9493	13549
146	F		Z	SEP1987	12046	16192
146	F		Z	OCT1987	12067	16266
146	F		Z	NOV1987	12558	17596
146	F		Z	DEC1987	12759	18301
146	F		Z	JAN1988	12642	18082
146	F		Z	FEB1988	12409	17470
158	F		Z	SEP1987	13841	16558
158	F		Z	OCT1987	13754	16499
158	F		Z	NOV1987	14222	17505
158	F		Z	DEC1987	14768	18423
158	F		Z	JAN1988	14933	18565
158	F		Z	FEB1988	14915	18331

Note that the data sets in Figure 14.1 and Table 14.1 are two different ways of representing the same data, namely foreign exchange rates for three different countries: the United Kingdom (COUNTRY='112'), Switzerland (COUNTRY='146') and Japan (COUNTRY='158'). The first representation (Figure 14.1) incorporates country names into the series names, while the second representation (Table 14.1) represents countries as different cross sections. See "Time Series and SAS Data Sets" in Chapter 2, "Working with Time Series Data."

Obtaining Descriptive Information on Cross Sections

If you want to know the unique set of values BY variables assume for each cross section in the data file, use the OUTBY= option. For example, the following statements list some of the cross sections available for an IFS file.

```
filename ifsfile 'host-specific-file-name' <host-options>;
proc datasource filetype=imfifsp infile=ifsfile
                interval=month outby=xsection;
run;

title1 'Some Cross Sections Available in IFSFILE';
proc print data=xsection noobs;
run;
```

```
                    Some Cross Sections Available in IFSFILE

                                     e
  c      p   v        s          n              n   n
  o      a   e        t          d              s   s  c
  u      r   r        _          _    n         e   e  _
  n      t   s        d          d    t    n    r   l  n
  t  c   n   i        a          a    i    o    i   e  a
  r  s   e   o        t          t    m    b    e   c  m
  y  c   r   n        e          e    e    s    s   t  e

  1  F  900  Z        .          .    .    0    0   0  WORLD
  1  F       Z  JAN1957  DEC1989  396  396  46  23  WORLD
  1  T       Z  JAN1957  DEC1989  396  396  16   8  WORLD
 10  F       Z  JAN1957  DEC1989  396  396  32  16  ALL COUNTRIES
 10  F  900  Z        .          .    .    0    0   0  ALL COUNTRIES
 10  M       Z  JAN1957  NOV1989  395  395   2   1  ALL COUNTRIES
 10  T       Z  JAN1957  DEC1989  396  396  18   9  ALL COUNTRIES
 16  F       Z  JAN1970  SEP1989  237  237  12   6  OFFSHORE BNKING CTRS
 16  F  900  Z        .          .    .    0    0   0  OFFSHORE BNKING CTRS
 24  F       Z  JAN1962  JUL1989  331  331   2   1  ACP COUNTRIES
```

Figure 14.4. Partial Listing of the OUTBY= Data Set

The OUTBY= data set reports the total number of series, NSERIES, defined in each cross section, NSELECT of which represent the selected variables. If you want to see the descriptive information on each of these NSELECT variables for each cross section, specify the OUTALL= option. For example, the following statements print descriptive information on the eight series defined for cross section (COUNTRY='1' CSC='T' PARTNER=' ' and VERSION='Z'):

```
filename ifsfile 'host-specific-file-name' <host-options>;
proc datasource filetype=imfifsp infile=ifsfile interval=month
                outall=ifsall;
run;

title1 'Time Series Defined in Cross Section';
title2 "COUNTRY='1'  CSC='T'  PARTNER=' '  VERSION='Z'";
proc print data=ifsall noobs;
   where country='1' and csc='T' and  partner=' ' and version='Z';
run;
```

```
                   Time Series Defined in Cross Section
              COUNTRY='1'  CSC='T'  PARTNER=' '  VERSION='Z'

                       s
c     p  v             e                                                 f
o     a  e             l     l  v   b                                 f  o
u     r  r             e     e  a   l  l                              o  r
n     t  s    n     k  c  t  n  r   k  a                              r  m
t  c  n  i    a     e  t  y  g  n   n  b                              m  a
r  s  e  o    m     p  e  p  t  u   u  e                              a  t
y  c  r  n    e     t  d  e  h  m   m  l                              t  l

1  T     Z  F__2KS  1  1  1  5  .  26  TOTAL PURCHASES                   0
1  T     Z  F__2LA  1  1  1  5  .  27  REPMTS.BY REPUR.IN PERIOD         0
1  T     Z  F__2MS  1  1  1  5  .  28  TOTAL PURCHASES BY OTHERS         0
1  T     Z  F__2NS  1  1  1  5  .  29  TOTAL REPURCHASES BY OTHERS       0
1  T     Z  F_C2KS  1  1  1  5  .  30  TOTAL PURCHASES,CUM.              0
1  T     Z  F_C2LA  1  1  1  5  .  31  REPAYMENTS BY REPURCHASE,CUM.     0
1  T     Z  F_C2MS  1  1  1  5  .  32  TOTAL PURCHASES BY OTHERS,CUM     0
1  T     Z  F_C2NS  1  1  1  5  .  33  TOTAL REP.BY OTHERS,CUM.          0

                  e                         d                          b
f        s        n                   s     a  d         d             a
o        t        d              c    u  s  t  u         u             s  s
r        _        _   n          _    b  c  a  _         _             e  o
m        d        d   t    n     n    j  d  t  c         n          n  y  u
a        a        a   i    o     a    e  a  y  o         a          d  e  r
t        t        t   m    b     m    c  t  p  d         m          e  a  c
d        e        e   e    s     e    t  a  e  e         e          c  r  e

0  JAN1957  DEC1989  396  396  WORLD        F  S  MILLIONS OF SDRS  1     T
0  JAN1957  DEC1989  396  396  WORLD        F  S  MILLIONS OF SDRS  2     T
0  JAN1957  DEC1989  396  396  WORLD        F  S  MILLIONS OF SDRS  1     T
0  JAN1957  DEC1989  396  396  WORLD        F  S  MILLIONS OF SDRS  2     T
0  JAN1957  NOV1986  359  359  WORLD     C  S  S  MILLIONS OF SDRS  1
0  JAN1957  DEC1989  396  396  WORLD     C  S  S  MILLIONS OF SDRS  1
0  JAN1957  NOV1986  359  359  WORLD     C  S  S  MILLIONS OF SDRS  1
0  JAN1957  DEC1989  396  396  WORLD     C  S  S  MILLIONS OF SDRS  1
```

Figure 14.5. Partial Listing of the OUTALL= Data Set

The OUTCONT= data set contains one observation for each time series variable with the descriptive information summarized over BY groups. When the data file contains no cross sections, the OUTCONT= and OUTALL= data sets are equivalent, except that the OUTALL= data set also reports time ranges for which data are available. The OUTBY= data set in this case contains a single observation reporting the number of series and time ranges for the whole data file.

Subsetting a Data File Containing Cross Sections

Data files containing cross sections can be subsetted by controlling which cross sections to include in the output data set. Selecting a subset of cross sections is accomplished using the WHERE statement. The WHERE statement gives a condition the BY variables must satisfy for a cross section to be selected. For example, the following statements extract the monthly effective exchange rate (F_X_AM) and exchange rate index (F_X_AF) for the United Kingdom (COUNTRY='112'), Switzerland (COUNTRY='146'), and Japan (COUNTRY='158') for the period from September, 1987 to February, 1988.

```
filename ifsfile 'host-specific-file-name' <host-options>;
proc datasource filetype=imfifsp infile=ifsfile interval=month
                out=exchange;
   where country in ('112','146','158') and partner=' ';
   keep  f_x_ah f_x_am;
   range from '01sep87'd to '01feb88'd;
run;

title1 'Printout of the OUT= Data Set';
proc print data=exchange noobs;
run;
```

Renaming Time Series Variables

Sometimes the time series variable names as given by data vendors are not descriptive enough, or you may prefer a different naming convention. In such cases, you can use the RENAME statement to assign more meaningful names to time series variables. You can also use LABEL statements to associate descriptive labels with your series variables.

For example, the series names for effective exchange rate (F_X_AM) and exchange rate index (F_X_AH) used by IMF can be given more descriptive names and labels by the following statements:

```
filename ifsfile 'host-specific-file-name' <host-options>;
proc datasource filetype=imfifsp infile=ifsfile interval=month
                out=exchange outcont=exchvars;
   where country in ('112','146','158') and partner=' ';
   keep  f_x_ah f_x_am;
   range from '01jun87'd to '01feb88'd;
   rename  f_x_ah=exrindex  f_x_am=effexr;
   label   f_x_ah='F_X_AH: Exchange Rate Index 1985=100'
           f_x_am='F_X_AM: Effective Exchange Rate(MERM)';
run;

title1 'Printout of OUTCONT= Showing New NAMEs and LABELs';
proc print data=exchvars noobs;
   var  name label length;
run;

title1 'Contents of OUT= Showing New NAMEs and LABELs';
proc contents data=exchange;
run;
```

The RENAME statement allows input names to be quoted strings. If the name of a series in the input file contains blanks or special characters that are not valid SAS name syntax, use the SAS option VALIDVARNAME= ANY or put the series name in quotes to rename it. See the FAME example using rename in the "Selecting Time Series Variables – The KEEP and DROP Statements" section (page 648).

```
                Printout of OUTCONT= Showing New NAMEs and LABELs

             name                       label                    length

           EFFEXR      F_X_AM: Effective Exchange Rate(MERM)       5
           EXRINDEX    F_X_AH: Exchange Rate Index 1985=100        5
```

```
                 Contents of OUT= Showing New NAMEs and LABELs

                             The CONTENTS Procedure

Data Set Name: WORK.EXCHANGE                          Observations:         27
Member Type:   DATA                                   Variables:            7
Engine:        V7                                     Indexes:              0
Created:       22:11 Saturday, May 30, 1998           Observation Length:   24
Last Modified: 22:11 Saturday, May 30, 1998           Deleted Observations: 0
Protection:                                           Compressed:           NO
Data Set Type:                                        Sorted:               NO
Label:


                  -----Engine/Host Dependent Information-----

   Data Set Page Size:         8192
   Number of Data Set Pages:   1
   First Data Page:            1
   Max Obs per Page:           338
   Obs in First Data Page:     27
   Number of Data Set Repairs: 0
   File Name:                  /tmp/SAS_work2C5200004EF6/exchange.sas7bdat
   Release Created:            7.00.00P
   Host Created:               HP-UX
   Inode Number:               9622
   Access Permission:          rw-r--r--
   Owner Name:                 sasknh
   File Size (bytes):          16384


             -----Alphabetic List of Variables and Attributes-----

 #  Variable  Type  Len  Pos  Format   Label
 ---------------------------------------------------------------------------
 3  country   Char    3    4           COUNTRY CODE
 4  csc       Char    1    7           CONTROL SOURCE CODE
 7  date      Num     4    0  MONYY7.  Date of Observation
 2  effexr    Num     5   17           F_X_AM: Effective Exchange Rate(MERM)
 1  exrindex  Num     5   12           F_X_AH: Exchange Rate Index 1985=100
 5  partner   Char    3    8           PARTNER COUNTRY CODE
 6  version   Char    1   11           VERSION CODE
```

Figure 14.6. Renaming and Labeling Variables

Notice that even though you changed the names of F_X_AH and F_X_AM to EXRINDEX and EFFEXR, respectively, you still used their old names in the KEEP and LABEL statements because renaming takes place at the output stage.

Changing the Lengths of Numeric Variables

The length attribute indicates the number of bytes the SAS System uses for storing the values of variables in output data sets. Therefore, the shorter the variable lengths, the more efficient the disk-space usage. However, there is a trade-off. The lengths of numeric variables are closely tied to their precision, and reducing their lengths arbitrarily can cause precision loss.

The DATASOURCE procedure uses default lengths for series variables appropriate to each file type. For example, the default lengths for numeric variables are 5 for IMFIFSP type files. In some cases, however, you may want to assign different lengths. Assigning lengths less than the defaults reduces memory and disk-space usage at the expense of reduced precision. Specifying lengths longer than the defaults increases the precision but causes the DATASOURCE procedure to use more memory and disk space. The following statements define a default length of 4 for all numeric variables in the IFSFILE and then assign a length of 6 to the exchange rate index:

```
filename ifsfile ’host-specific-file-name’ <host-options>;
proc datasource filetype=imfifsp infile=ifsfile interval=month
                out=exchange outcont=exchvars;
   where country in (’112’,’146’,’158’) and partner=’   ’;
   keep  f_x_am f_x_ah;
   range from ’01jun87’d to ’01feb88’d;
   rename  f_x_ah=exrindex  f_x_am=effexr;
   label   f_x_ah=’F_X_AH: Exchange Rate Index 1985=100’
           f_x_am=’F_X_AM: Effective Exchange Rate(MERM)’;
   length _numeric_ 4; length f_x_ah 6;
run;

title1 ’Printout of OUTCONT= Showing LENGTH Variable’;
proc print data=exchvars noobs;
   var  name label length;
run;

title1 ’Contents of the OUT= Data Set Showing LENGTHs’;
proc contents data=exchange;
run;
```

```
                  Printout of OUTCONT= Showing LENGTH Variable

      name                        label                        length

    EFFEXR      F_X_AM: Effective Exchange Rate(MERM)            4
    EXRINDEX    F_X_AH: Exchange Rate Index 1985=100             6
```

```
                 Contents of the OUT= Data Set Showing LENGTHs

                             The CONTENTS Procedure

Data Set Name: WORK.EXCHANGE                          Observations:         27
Member Type:   DATA                                   Variables:            7
Engine:        V7                                     Indexes:              0
Created:       22:11 Saturday, May 30, 1998           Observation Length:   24
Last Modified: 22:11 Saturday, May 30, 1998           Deleted Observations: 0
Protection:                                           Compressed:           NO
Data Set Type:                                        Sorted:               NO
Label:


                  -----Engine/Host Dependent Information-----

   Data Set Page Size:         8192
   Number of Data Set Pages:   1
   First Data Page:            1
   Max Obs per Page:           338
   Obs in First Data Page:     27
   Number of Data Set Repairs: 0
   File Name:                  /tmp/SAS_work2C5200004EF6/exchange.sas7bdat
   Release Created:            7.00.00P
   Host Created:               HP-UX
   Inode Number:               9628
   Access Permission:          rw-r--r--
   Owner Name:                 sasknh
   File Size (bytes):          16384


             -----Alphabetic List of Variables and Attributes-----

 #  Variable  Type  Len  Pos  Format   Label
 ---------------------------------------------------------------------------
 3  country   Char    3    8           COUNTRY CODE
 4  csc       Char    1   11           CONTROL SOURCE CODE
 7  date      Num     4    4  MONYY7.  Date of Observation
 2  effexr    Num     4    0           F_X_AM: Effective Exchange Rate(MERM)
 1  exrindex  Num     6   16           F_X_AH: Exchange Rate Index 1985=100
 5  partner   Char    3   12           PARTNER COUNTRY CODE
 6  version   Char    1   15           VERSION CODE
```

Figure 14.7. Changing the Lengths of Numeric Variables

The default lengths of the character variables are set to the minimum number of characters that can hold the longest possible value.

Syntax

The DATASOURCE procedure uses the following statements:

> **PROC DATASOURCE** *options***;**
>   **KEEP** *variable-list***;**
>   **DROP** *variable-list***;**
>   **KEEPEVENT** *event-list***;**
>   **DROPEVENT** *event-list***;**
>   **WHERE** *where-expression***;**
>   **RANGE FROM** *from* **TO** *to***;**
>   **ATTRIBUTE** *variable-list attribute-list* . . . **;**
>   **FORMAT** *variable-list format* . . . **;**
>   **LABEL** *variable="label"* . . . **;**
>   **LENGTH** *variable-list length* . . . **;**
>   **RENAME** *old-name=new-name* . . . **;**

The PROC DATASOURCE statement is required. All the rest of the statements are optional.

The DATASOURCE procedure uses two kinds of statements:

1. subsetting statements, which control what time series, time periods, and cross sections are extracted from the input data file
2. attribute statements, which control the attributes of the variables in the output SAS data set

The subsetting statements are the KEEP, DROP, KEEPEVENT, and DROPEVENT statements (which select output variables); the RANGE statement (which selects time ranges); and the WHERE statement (which selects cross sections). The attribute statements are the ATTRIBUTE, FORMAT, LABEL, LENGTH, and RENAME statements.

The statements and options used by PROC DATASOURCE are summarized in Table 14.2.

Table 14.2. Summary of Syntax

Description	Statement	Option
Input Data File Options		
specify the character set of the incoming	PROC DATASOURCE	ASCII
data	PROC DATASOURCE	EBCDIC

Description	Statement	Option
specify the type of input data file to read	PROC DATASOURCE	FILETYPE=
specify the fileref(s) of the input data file(s)	PROC DATASOURCE	INFILE=
specify the lrecl(s) of the input data files(s)	PROC DATASOURCE	LRECL=
specify the recfm(s) of the input data files(s)	PROC DATASOURCE	RECFM=
Output Data Set Options		
write the extracted time series data	PROC DATASOURCE	OUT=
output the descriptive information on the time series variables and cross sections	PROC DATASOURCE	OUTALL=
output the descriptive information on the cross sections	PROC DATASOURCE	OUTBY=
output the descriptive information on the time series variables	PROC DATASOURCE	OUTCONT=
write event-oriented data	PROC DATASOURCE	OUTEVENT=
control whether all or only selected series and cross sections be reported	PROC DATASOURCE	OUTSELECT=
create single indexes from BY variables for the OUT= data set	PROC DATASOURCE	INDEX
control the alignment of SAS Date values	PROC DATASOURCE	ALIGN=
Subsetting		
specify the periodicity of series to be extracted	PROC DATASOURCE	INTERVAL=
specify the time series variables to be included in the OUT= data set	KEEP	
specify the time series variables to be excluded from the OUT= data set	DROP	
specify the events to be included in the OUTEVENT= data set	KEEPEVENT	
specify the events to be excluded from the OUTEVENT= data set	DROPEVENT	
select cross sections for output	WHERE	
specify the time range of observations to be output	RANGE	

Description	**Statement**	**Option**
Assigning Attributes		
assign formats to the output variables	FORMAT	
	ATTRIBUTE	FORMAT=
assign labels to variables in the output data sets	LABEL	
	ATTRIBUTE	LABEL=
control the lengths of the output variables	LENGTH	
	ATTRIBUTE	LENGTH=
assign new names to the output variables	RENAME	

PROC DATASOURCE Statement

PROC DATASOURCE *options;*

The following options can be used in the PROC DATASOURCE statement:

ALIGN= *option*
: controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

ASCII
: specifies the incoming data is ascii. This option is needed when the native character set of your host machine is ebcdic.

DBNAME= **'***database name***'**
: specifies the FAME database to access. Only use this option with the filetype=FAME option. The character string you specify on the DBNAME= option is passed through to FAME. Specify the value of this option as you would in accessing the database from within FAME software.

EBCDIC
: specifies the incoming data is ebcdic. This option is needed when the native character set of your host machine is ascii.

FAMEPRINT
: prints the FAME command file generated by PROC DATASOURCE and the log file produced by the FAME component of the interface system. Only use this option with the filetype=FAME option.

FILETYPE= *entry*
DBTYPE= *dbtype*
: specifies the kind of input data file to process. See the "Supported File Types" section on page 677 for a list of supported file types. The FILETYPE= option is required.

INDEX
: creates a set of single indexes from BY variables for the OUT= data set. Under some circumstances, creating indexes for a SAS data set may increase the efficiency in

locating observations when BY or WHERE statements are used in subsequent steps. Refer to *SAS Language: Reference, Version 7, First Edition* for more information on SAS indexes. The INDEX option is ignored when no OUT= data set is created or when the data file does not contain any BY variables. The INDEX= data set option can be used to override the index variable definitions.

INFILE= *fileref*
INFILE= *(fileref1 fileref2 ... filerefn)*
specifies the *fileref* assigned to the input data file. The default value is DATAFILE. The fileref used in INFILE= option (or if no INFILE= option is specified, the fileref DATAFILE) must be associated with the physical data file in a FILENAME statement. (On some operating systems, the fileref assignment can be made with the system's control language, and a FILENAME statement may not be needed. Refer to *SAS Language: Reference Version 7, First Edition* for more details on the FILENAME statement). Physical data files can reside on tapes, disks, diskettes, CD-ROM, or other media.

For some file types, the data are distributed over several files. In this case, the INFILE= option is required, and it lists in parentheses the filerefs for each of the files making up the database. The order in which these FILEREFS are listed is important and must conform to the specifics of each file type as explained in the "Supported File Types" section on page 677.

LRECL= *lrecl*
LRECL= *(lrecl1 lrecl2 ... lrecln)*
The logical record length in bytes of the infile. Only use this if you need to override the default LRECL of the file. For some file types, the data are distributed over several files. In this case, the LRECL= option lists in parentheses the LRECLS for each of the files making up the database. The order in which these lrecls are listed is important and must conform to the specifics of each file type as explained in the "Supported File Types" section on page 677.

RECFM= *recfm*
RECFM= *(recfm1 recfm2 ... recfmn)*
The record format of the infile. Only use this if you need to override the default record format of the file. For some file types, the data are distributed over several files. In this case, the RECFM= option lists in parentheses the recfms for each of the files making up the database. The order in which these RECFMS are listed is important and must conform to the specifics of each file type as explained in the "Supported File Types" section on page 677. The possible values of RECFM are:

- F or FIXED for fixed length records
- N or BIN for binary records
- D or VAR for varying length records
- U or DEF for host default record format
- DOM_V or DOMAIN_VAR or BIN_V or BIN_VAR for unix binary record format

INTERVAL= *interval*
FREQUENCY= *interval*
TYPE= *interval*

specifies the periodicity of series selected for output to the OUT= data set. The OUT= data set created by PROC DATASOURCE can contain only time series with the same periodicity. Some data files contain time series with different periodicities; for example, a file may contain both monthly series and quarterly series. Use the INTERVAL= option to indicate which periodicity you want. If you want to extract series with different periodicities, use different PROC DATASOURCE invocations with the desired INTERVAL= options.

Common values for INTERVAL= are YEAR, QUARTER, MONTH, WEEK, and DAY. The values allowed, as well as the default value of the INTERVAL= option, depend on the file type. See the "Supported File Types" section on page 677 for the INTERVAL= values appropriate to the data file type you are reading.

OUT= *SAS-data-set*

names the output data set for the time series extracted from the data file. If none of the output data set options are specified, including the OUT= data set itself, an OUT= data set is created and named according to the DATA*n* convention. However, when you create any of the other output data sets, such as OUTCONT=, OUTBY=, OUTALL=, or OUTEVENT=, you must explicitly specify the OUT= data set; otherwise, it will not be created. See the "OUT= Data Set" section on page 672 for further details.

OUTALL= *SAS-data-set*

writes information on the contents of the input data file to an output data set. The OUTALL= data set includes descriptive information, time ranges, and observation counts for all the time series within each BY group. By default, no OUTALL= data set is created.

The OUTALL= data set contains the Cartesian product of the information output by the OUTCONT= and OUTBY= options. In data files for which there are no cross sections, the OUTALL= and OUTCONT= data sets are almost equivalent, except that OUTALL= data set also reports time ranges and observation counts of series. See the "OUTALL= Data Set" section on page 675 for further details.

OUTBY= *SAS-data-set*

writes information on the BY variables to an output data set. The OUTBY= data set contains the list of cross sections in the database delimited by the unique set of values that the BY variables assume. Unless the OUTSELECT=OFF option is present, only the selected BY groups get written to the OUTBY= data set. If you omit the OUTBY= option, no OUTBY= data set is created. See the "OUTBY= Data Set" section on page 674 for further details.

OUTCONT= *SAS-data-set*

writes information on the contents of the input data file to an output data set. By default, the OUTCONT= data set includes descriptive information on all of the unique series of the selected periodicity in the data file. When the OUTSELECT=OFF option is omitted, the OUTCONT= data set includes observations only for the series selected for output to the OUT= data set. By default, no OUTCONT= data set is created. See the "OUTCONT= Data Set" section on page 673 for further details.

OUTEVENT= *SAS-data-set*
names the output data set to output event-oriented time series data. This option can only be used when CRSP stock files are being processed. For all other file types, it will be ignored. See the "OUTEVENT= Data Set" section on page 676 for further details.

OUTSELECT= ON | OFF
determines whether to output all observations (OUTSELECT=OFF) or only those corresponding to the selected time series and selected BY groups (OUTSELECT=ON) to OUTCONT=, OUTBY=, and OUTALL= data sets. The default is OUTSELECT=ON. The OUTSELECT= option is only relevant when any one of the auxiliary data sets is specified. The option writes observations to OUTCONT=, OUTBY=, and OUTALL= data sets for only the selected time series and selected BY groups if it is set ON. The OUTSELECT= option is only relevant when any one of the OUTCONT=, OUTBY= and OUTALL= options are specified. The default is OUTSELECT=ON.

KEEP Statement

KEEP *variable-list;*

The KEEP statement specifies which variables in the data file are to be included in the OUT= data set. Only the time series and event variables can be specified in a KEEP statement. All the BY variables and the time ID variable DATE are always included in the OUT= data set; they cannot be referenced in a KEEP statement. If they are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See the section "Variable Lists" on page 670 for details.

There is a default KEEP list for each file type. Usually, descriptor type variables, like footnotes, are not included in the default KEEP list. If you give a KEEP statement, the default list becomes undefined.

Only one KEEP or one DROP statement can be used. KEEP and DROP are mutually exclusive.

You can also use the KEEP= data set option to control which variables to include in the OUT= data set. However, the KEEP statement differs from the KEEP= data set option in several aspects:

- The KEEP statement selection is applied before variables are read from the data file, while the KEEP= data set option selection is applied after variables are read and as they are written to the OUT= data set. Therefore, using the KEEP statement instead of the KEEP= data set option is much more efficient.
- If the KEEP statement causes no series variables to be selected, then no observations are output to the OUT= data set.

- The KEEP statement variable specifications are applied to each cross section independently. This behavior may produce different variables than those produced by the KEEP= data set option when order-range variable list specifications are used.

DROP Statement

DROP *variable-list;*

The DROP statement specifies that some variables be excluded from the OUT= data set. Only the time series and event variables can be specified in a DROP statement. None of the BY variables or the time ID variable DATE can be excluded from the OUT= data set. If they are referenced in a DROP statement, a warning message is given and the reference is ignored. Use the WHERE statement for selection based on BY variables, and use the RANGE statement for date selections.

The variable list can contain variable names or name range specifications. See the section "Variable Lists" on page 670 for details.

Only one DROP or one KEEP statements can be used. KEEP and DROP are mutually exclusive.

There is a default KEEP list for each file type. Usually, descriptor type variables, like footnotes, are not included in the default KEEP list. If you specify a DROP statement, the default list becomes undefined.

You can also use the DROP= data set option to control which variables to exclude from the OUT= data set. However, the DROP statement differs from the DROP= data set option in several aspects:

- The DROP statement selection is applied before variables are read from the data file, while the DROP= data set option selection is applied after variables are read and as they are written to the OUT= data set. Therefore, using the DROP statement instead of the DROP= data set option is much more efficient.
- If the DROP statement causes all series variables to be excluded, then no observations are output to the OUT= data set.
- The DROP statement variable specifications are applied to each cross section independently. This behavior may produce different variables than those produced by the DROP= data set option when order-range variable list specifications are used.

KEEPEVENT Statement

KEEPEVENT *variable-list;*

The KEEPEVENT statement specifies which event variables in the data file are to be included in the OUTEVENT= data set. As a result, the KEEPEVENT statement is valid only for data files containing event-oriented time series data, that is, only for CRSP files. All the BY variables, the time ID variable DATE and the event-grouping variable EVENT are always included in the OUTEVENT= data set. These variables can not be referenced in the KEEPEVENT statement. If any of these variables are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See the section "Variable Lists" on page 670 for details.

Only one KEEPEVENT or one DROPEVENT statement can be used. KEEPEVENT and DROPEVENT are mutually exclusive.

You can also use the KEEP= data set option to control which event variables to include in the OUTEVENT= data set. However, the KEEPEVENT statement differs from the KEEP= data set option in several aspects:

- The KEEPEVENT statement selection is applied before variables are read from the data file, while the KEEP= data set option selection is applied after variables are read and as they are written to the OUTEVENT= data set. Therefore, using the KEEPEVENT statement instead of the KEEP= data set option is much more efficient.
- If the KEEPEVENT statement causes no event variables to be selected, then no observations are output to the OUTEVENT= data set.

DROPEVENT Statement

DROPEVENT *variable-list;*

The DROPEVENT statement specifies that some event variables be excluded from the OUTEVENT= data set. As a result, the DROPEVENT statement is valid only for data files containing event-oriented time series data, that is, only for CRSP files. All the BY variables, the time ID variable DATE, and the event-grouping variable EVENT are always included in the OUTEVENT= data set. These variables cannot be referenced in the DROPEVENT statement. If any of these variables are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See the section "Variable Lists" on page 670 for details.

Only one DROPEVENT or one KEEPEVENT statement can be used. DROPEVENT and KEEPEVENT are mutually exclusive.

You can also use the DROP= data set option to control which event variables to exclude from the OUTEVENT= data set. However, the DROPEVENT statement differs from the DROP= data set option in several aspects:

- The DROPEVENT statement selection is applied before variables are read from the data file, while the DROP= data set option selection is applied after variables are read and as they are written to the OUTEVENT= data set. Therefore, using the DROPEVENT statement instead of the DROP= data set option is much more efficient.
- If the DROPEVENT statement causes all series variables to be excluded, then no observations are output to the OUTEVENT= data set.

WHERE Statement

WHERE *where-expression;*

The WHERE statement specifies conditions that BY variables must satisfy in order for a cross section to be included in the OUT= and OUTEVENT= data sets. By default, all BY groups are selected.

The *where-expression* must refer only to BY variables defined for the file type you are reading. The section "Supported File Types" on page 677 lists the names of the BY variables for each file type.

For example, DOTS (Direction of Trade Statistics) files, distributed by International Monetary Fund, have four BY variables: COUNTRY, CSC, PARTNER, and VERSION. Both COUNTRY and PARTNER are three-digit country codes. To select the direction of trade statistics of the United States (COUNTRY='111') with Turkey (COUNTRY='186'), Japan (COUNTRY='158'), and the oil exporting countries group (COUNTRY='985'), you should specify

```
where country='111' and partner in ('186','158','985');
```

You can use any SAS language operators and special WHERE expression operators in the WHERE statement condition. Refer to *SAS Language: Reference, Version 7, First Edition* for a more detailed discussion of WHERE expressions.

If you want to see the names of the BY variables and the values they assume for each cross section, you can first run PROC DATASOURCE with only the OUTBY= option. The information contained in the OUTBY= data set will aid you in selecting the appropriate BY groups for subsequent PROC DATASOURCE steps.

RANGE Statement

RANGE FROM *from* **TO** *to;*

The RANGE statement selects the time range of observations written to the OUT= and OUTEVENT= data sets. The *from* and *to* values can be SAS date, time, or datetime constants, or they can be specified as *year* or *year***:***period*, where *year* is a two-digit or four-digit year, and *period* (when specified) is a period within the year corresponding to the INTERVAL= option. (For example, if INTERVAL=QTR, then *period* refers to quarters.) When *period* is omitted, the beginning of the year is assumed for the *from* value, and the end of the year is assumed for the *to* value.

If a 2-digit year is specified, PROC DATASOURCE complies to year 2000 guidelines by using the current value of the YEARCUTOFF option to determine the century of your data. Warnings are issued in the SAS log whenever DATASOURCE needs to determine the century from a 2-digit year specification.

The default YEARCUTOFF is 1920. To use a different yearcutoff, specify

```
options yearcutoff=yyyy;
```

where yyyy is the yearcutoff you want to use. See the *SAS Language: Reference, Version 7, First Edition* for a more detailed discussion of the YEARCUTOFF option.

Both the FROM and TO specifications are optional, and both the FROM and TO keywords are optional. If the FROM limit is omitted, the output observations start with the minimum date for which data is available for any selected series. Similarly, if the TO limit is omitted, the output observations end with the maximum date for which data are available.

The following are some examples of RANGE statements:

```
range from 1980 to 1990;
range 1980 - 1990;
range from 1980;
range 1980;
range to 1990;
range to 1990:2;
range from ’31aug89’d to ’28feb1990’d;
```

The RANGE statement applies to each BY group independently. If all the selected series contain no data in the specified range for a given BY group, then there will be no observations for that BY group in the OUT= and OUTEVENT= data sets.

If you want to know the time ranges for which periodic time series data is available, you can first run PROC DATASOURCE with the OUTBY= or OUTALL= options. OUTBY= data set reports the union of the time ranges over all the series within each BY group, while the OUTALL= data set gives time ranges for each series separately in each BY group.

ATTRIBUTE Statement

ATTRIBUTE *variable-list attribute-list ... ;*

The ATTRIBUTE statement assigns formats, labels, and lengths to variables in the output data sets.

The *variable-list* can contain variable names and variable name range specifications. See the section “Variable Lists” on page 670 for details. The attributes specified in the following attribute list apply to all variables in the variable list:

An *attribute-list* consists of one or more of the following options:

FORMAT= *format*
associates a format with variables in *variable-list*. The *format* can be either a standard SAS format or a format defined with the FORMAT procedure. The default formats for variables depend on the file type.

LABEL= *"label"*
assigns a label to the variables in the variable list. The default labels for variables depend on the file type. Labels can be up to 256 bytes in length.

LENGTH= *length*
specifies the number of bytes used to store the values of variables in the variable list.

The default lengths for numeric variables depend on the file type. Usually default lengths are set to 5 bytes. (For CRSP files, the default lengths are 6 bytes).

The length specification also controls the amount of memory that PROC DATASOURCE uses to hold variable values while processing the input data file. Thus, specifying a LENGTH= value smaller than the default will reduce both the disk space taken up by the output data sets and the amount of memory used by the PROC DATASOURCE step, at the cost of reduced precision of output data values.

FORMAT Statement

FORMAT *variable-list format ... ;*

The FORMAT statement assigns formats to variables in output data sets. The *variable-list* can contain variable names and variable name range specifications. See the section "Variable Lists" on page 670 for details. The format specified applies to all variables in the variable list.

A single FORMAT statement can assign the same format to several variables or different formats to different variables. The FORMAT statement can use standard SAS formats or formats defined using the FORMAT procedure.

Any later format specification for a variable, using either the FORMAT statement or the FORMAT= option in the ATTRIBUTE statement, always overrides the previous one.

LABEL Statement

LABEL *variable = "label" ... ;*

The LABEL statement assigns SAS variable labels to variables in the output data sets. You can give labels for any number of variables in a single LABEL statement. The default labels for variables depend on the file type. Extra long labels (> 256 bytes) reside in the OUTCONT data set as the DESCRIPT variable.

Any later label specification for a variable, using either the LABEL statement or the LABEL= option in the ATTRIBUTE statement, always overrides the previous one.

LENGTH Statement

LENGTH *variable-list length ... ;*

The LENGTH statement, like the LENGTH= option in the ATTRIBUTE statement, specifies the number of bytes used to store values of variables in output data sets. The default lengths for numeric variables depend on the file type. Usually default lengths are set to 5 bytes. (For CRSP files, the default lengths are 6 bytes).

The default lengths of character variables are defined as the minimum number of characters that can hold the longest possible value.

For some file types, the LENGTH statement also controls the amount of memory used to store values of numeric variables while processing the input data file. Thus, specifying LENGTH values smaller than the default will reduce both the disk space taken up by the output data sets and the amount of memory used by the PROC DATASOURCE step, at the cost of reduced precision of output data values.

Any later length specification for a variable, using either the LENGTH statement or the LENGTH= option in the ATTRIBUTE statement, always overrides the previous one.

RENAME Statement

RENAME *old-name = new-name ... ;*

The RENAME statement is used to change the names of variables in the output data sets. Any number of variables can be renamed in a single RENAME statement. The most recent RENAME specification overrides any previous ones for a given variable. The new-name is limited to thirty-two characters.

Renaming of variables is done at the output stage. Therefore, you need to use the old variable names in all other PROC DATASOURCE statements. For example, the series variable names DATA1-DATA350 used with annual COMPUSTAT files are not very descriptive, so you may choose to rename them to reflect the financial aspect they represent. You may rename "DATA51" to "INVESTTAX" with the RENAME statement

```
rename data51=investtax;
```

since it contains investment tax credit data. However, in all other DATASOURCE statements, you must use the old name, DATA51.

Details

Variable Lists

Variable lists used in PROC DATASOURCE statements can consist of any combination of variable names and name range specifications. Items in variable lists can have the following forms:

- a name, for example, PZU.
- an alphabetic range *name1-name2*. For example, A-DZZZZZZZ specifies all variables with names starting with A, B, C, or D.
- a prefix range *prefix*:. For example, IP: selects all variables with names starting with the letters IP.
- an order range *name1--name2*. For example, GLR72--GLRD72 specifies all variables in the input data file between GLR72 and GRLD72 inclusive.
- a numeric order range *name1*-NUMERIC-*name2*. For example, GLR72-NUMERIC-GLRD72 specifies all numeric variables between GLR72 and GRLD72 inclusive.

- a character order range *name1*-CHARACTER-*name2*. For example, GLR72-CHARACTER-GLRD72 specifies all character variables between GLR72 and GRLD72 inclusive.
- one of the keywords _NUMERIC_, _CHARACTER_, or _ALL_. _NUMERIC_ specifies all numeric variables. _CHARACTER_ specifies all character variables. _ALL_ specifies all variables.

To determine the order of series in a data file, run PROC DATASOURCE with the OUTCONT= option, and print the output data set. Note that order and alphabetic range specifications are inclusive, meaning that the beginning and ending names of the range are also included in the variable list.

For order ranges, the names used to define the range must actually name variables in the input data file. For alphabetic ranges, however, the names used to define the range need not be present in the data file.

Note that variable specifications are applied to each cross section independently. This may cause the order-range variable list specification to behave differently than its DATA step and data set option counterparts. This is because PROC DATASOURCE knows which variables are defined for which cross sections, while the DATA step applies order range specification to the whole collection of time series variables.

If the ending variable name in an order range specification is not in the current cross section, all variables starting from the beginning variable to the last variable defined in that cross section get selected. If the first variable is not in the current cross section, then order range specification has no effect for that cross section.

The variable names used in variable list specifications can refer to either series names appearing in the input data file or to the SAS names assigned to series data fields internally if the series names are not recorded to the INFILE= file. When the latter is the case, internally defined variable names are listed in the section "Data Files" later in this chapter.

The following are examples of the use of variable lists:

```
keep  ip: pw112-pw117 pzu;
drop  data1-data99  data151-data350;
length data1-numeric-aftnt350  ucode 4;
```

The first statement keeps all the variables starting with IP:, all the variables between PW112 and PW117 including the PW112 and PW117 themselves, and a single variable PZU. The second statement drops all the variables that fall alphabetically between DATA1 and DATA99, and DATA151 and DATA350. Finally, the third statement assigns a length of 4 bytes to all the numeric variables defined between DATA1 and AFTNT350, and UCODE.

OUT= Data Set

The OUT= data set can contain the following variables:

- the BY variables, which identify cross-sectional dimensions when the input data file contains time series replicated for different values of the BY variables. Use the BY variables in a WHERE statement to process the OUT= data set by cross sections. The order in which BY variables are defined in the OUT= data set corresponds to the order in which the data file is sorted.
- DATE, a SAS date-, time-, or datetime- valued variable that reports the time period of each observation. The values of the DATE variable may span different time ranges for different BY groups. The format of the DATE variable depends on the INTERVAL= option.
- the periodic time series variables, which are included in the OUT= data set only if they have data in at least one selected BY group and they are not discarded by a KEEP or DROP statement
- the event variables, which are included in the OUT= data set if they are not discarded by a KEEP or DROP statement. By default, these variables are not output to OUT= data set.

The values of BY variables remain constant in each cross section. Observations within each BY group correspond to the sampling of the series variables at the time periods indicated by the DATE variable.

You can create a set of single indexes for the OUT= data set by using the INDEX option, provided there are BY variables. Under some circumstances, this may increase the efficiency of subsequent PROC and DATA steps that use BY and WHERE statements. However, there is a cost associated with creation and maintenance of indexes. The *SAS Language: Reference, Version 7, First Edition* lists the conditions under which the benefits of indexes outweigh the cost.

With data files containing cross sections, there can be various degrees of overlap among the series variables. One extreme is when all the series variables contain data for all the cross sections. In this case, the output data set is very compact. In the other extreme case, however, the set of time series variables are unique for each cross section, making the output data set very sparse, as depicted in Figure 14.8.

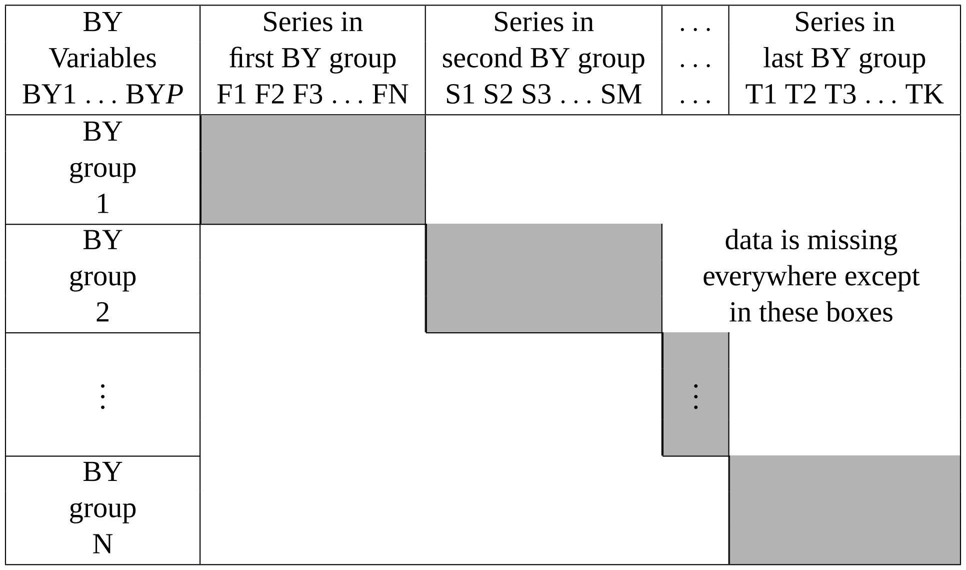

BY Variables BY1 . . . BY*P*
Series in first BY group F1 F2 F3 . . . FN
Series in second BY group S1 S2 S3 . . . SM
. . .
Series in last BY group T1 T2 T3 . . . TK
BY group 1
BY group 2
⋮
BY group N
data is missing everywhere except in these boxes

Figure 14.8. The OUT= Data Set containing unique Series for each BY Group

The data in Figure 14.8 can be represented more compactly if cross-sectional information is incorporated into series variable names.

OUTCONT= Data Set

The OUTCONT= data set contains descriptive information for the time series variables. This descriptive information includes various attributes of the time series variables. The OUTCONT= data set contains the following variables:

- NAME, a character variable that contains the series name.
- KEPT, a numeric variable that indicates whether the series was selected for output by the DROP or KEEP statements, if any. KEPT will usually be the same as SELECTED, but may differ if a WHERE statement is used.
- SELECTED, a numeric variable that indicates whether the series is selected for output to the OUT= data set. The series is included in the OUT= data set (SELECTED=1) if it is kept (KEPT=1) and it has data for at least one selected BY group.
- TYPE, a numeric variable that indicates the type of the time series variable. TYPE=1 for numeric series; TYPE=2 for character series.
- LENGTH, a numeric variable that gives the number of bytes allocated for the series variable in the OUT= data set.
- VARNUM, a numeric variable that gives the variable number of the series in the OUT= data set. If the series variable is not selected for output (SELECTED=0), then VARNUM has a missing value. Likewise, if no OUT= option is given, VARNUM has all missing values.
- LABEL, a character variable that contains the label of the series variable. LABEL contains only the first 256 characters of the labels. If they are longer

than 256 characters, then the variable, DESCRIPT, is defined to hold the whole length of series labels. Note that if a data file assigns different labels to the same series variable within different cross sections, only the first occurrence of labels will be transferred to the LABEL column.

- the variables FORMAT, FORMATL, and FORMATD, which give the format name, length, and number of format decimals, respectively.
- the GENERIC variables, whose values may vary from one series to another, but whose values remain constant across BY groups for the same series.

By default, the OUTCONT= data set contains observations for only the selected series, that is, for series where SELECTED=1. If the OUTSELECT=OFF option is specified, the OUTCONT= data set contains one observation for each unique series of the specified periodicity contained in the input data file.

If you do not know what series are in the data file, you can run PROC DATASOURCE with the OUTCONT= option and OUTSELECT=OFF. The information contained in the OUTCONT= data set can then help you to determine which time series data you want to extract.

OUTBY= Data Set

The OUTBY= data set contains information on the cross sections contained in the input data file. These cross sections are represented as BY groups in the OUT= data set. The OUTBY= data set contains the following variables:

- the BY variables, whose values identify the different cross sections in the data file. The BY variables depend on the file type.
- BYSELECT, a numeric variable that reports the outcome of the WHERE statement condition for the BY variable values for this observation. The value of BYSELECT is 1 for BY groups selected by the WHERE statement for output to the OUT= data set and is 0 for BY groups that are excluded by the WHERE statement. BYSELECT is added to the data set only if a WHERE statement is given. When there is no WHERE statement, then all the BY groups are selected.
- ST_DATE, a numeric variable that gives the starting date for the BY group. The starting date is the earliest of the starting dates of all the series that have data for the current BY group.
- END_DATE, a numeric variable that gives the ending date for the BY group. The ending date is the latest of the ending dates of all the series that have data for the BY group.
- NTIME, a numeric variable that gives the number of time periods between ST_DATE and END_DATE, inclusive. Usually, this is the same as NOBS, but they may differ when time periods are not equally spaced and when the OUT= data set is not specified. NTIME is a maximum limit on NOBS.
- NOBS, a numeric variable that gives the number of time series observations in OUT= data set between ST_DATE and END_DATE, inclusive. When a given

BY group is discarded by a WHERE statement, the NOBS variable corresponding to this BY group becomes 0, since the OUT= data set does not contain any observations for this BY group. Note that BYSELECT=0 for every discarded BY group.

- NINRANGE, a numeric variable that gives the number of observations in the range (*from,to*) defined by the RANGE statement. This variable is only added to the OUTBY= data set when the RANGE statement is specified.
- NSERIES, a numeric variable that gives the total number of unique time series variables having data for the BY group.
- NSELECT, a numeric variable that gives the total number of selected time series variables having data for the BY group.
- the generic variables, whose values remain constant for all the series in the current BY group.

In this list, you can only control the attributes of the BY and GENERIC variables.

The variables NOBS, NTIME, and NINRANGE give observation counts, while the variables NSERIES and NSELECT give series counts.

By default, observations for only the selected BY groups (where BYSELECT=1) are output to the OUTBY= data set, and the date and time range variables are computed over only the selected time series variables. If the OUTSELECT=OFF option is specified, the OUTBY= data set contains an observation for each BY group, and the date and time range variables are computed over all the time series variables.

For file types that have no BY variables, the OUTBY= data set contains one observation giving ST_DATE, END_DATE, NTIME, NOBS, NINRANGE, NSERIES, and NSELECT for all the series in the file.

If you do not know the BY variable names or their possible values, you can do an initial run of PROC DATASOURCE with the OUTBY= option. The information contained in the OUTBY= data set can help you design your WHERE expression and RANGE statement for the subsequent executions of PROC DATASOURCE to obtain different subsets of the same data file.

OUTALL= Data Set

The OUTALL= data set combines and expands the information provided by the OUTCONT= and OUTBY= data sets. That is, the OUTALL= data set not only reports the OUTCONT= information separately for each BY group, but also reports the OUTBY= information separately for each series. Each observation in the OUTBY= data set gets expanded to NSERIES or NSELECT observations in the OUTALL= data set, depending on whether the OUTSELECT=OFF option is specified.

By default, only the selected BY groups and series are included in the OUTALL= data set. If the OUTSELECT=OFF option is specified, then all the series within all the BY groups are reported.

The OUTALL= data set contains all the variables defined in the OUTBY= and OUTCONT= data sets and also contains the GENERIC variables (whose values may vary from one series to another and also from one BY group to another). Another additional variable is BLKNUM, which gives the data block number in the data file containing the series variable.

The OUTALL= data set is useful when BY groups do not contain the same time series variables or when the time ranges for series change across BY groups.

You should be careful in using the OUTALL= option, since the OUTALL= data set can get very large for many file types. Some file types have the same series and time ranges for each BY group; the OUTALL= option should not be used with these file types. For example, you should not specify the OUTALL= option with COMPUSTAT files, since all the BY groups contain the same series variables.

The OUTALL= and OUTCONT= data sets are equivalent when there are no BY variables, except that the OUTALL= data set contains extra information about the time ranges and observation counts of the series variables.

OUTEVENT= Data Set

The OUTEVENT= data set is used to output event-oriented time series data. Events occurring at discrete points in time are recorded along with the date they occurred. Only CRSP stock files contain event-oriented time series data. For all other types of files, the OUTEVENT= option is ignored.

The OUTEVENT= data set contains the following variables:

- the BY variables, which identify cross-sectional dimensions when the input data file contains time series replicated for different values of the BY variables. Use the BY variables in a WHERE statement to process the OUTEVENT= data set by cross sections. The order in which BY variables are defined in the OUTEVENT= data set corresponds to the order in which the data file is sorted.
- DATE, a SAS date-, time- or datetime- valued variable that reports the discrete time periods at which events occurred. The format of the DATE variable depends on the INTERVAL= option, and should accurately report the date based on the SAS YEARCUTOFF option. The default value for YEARCUTOFF is 1920. The dates used may span up to 250 years.
- EVENT, a character variable that contains the event group name. The EVENT variable is another cross-sectional variable.
- the event variables, included in the OUTEVENT= data set only if they have data in at least one selected BY group, are not discarded by a KEEPEVENT or DROPEVENT statement.

Note that each event group contains a nonoverlapping set of event variables; therefore, the OUTEVENT= data set is very sparse. You should exercise care when selecting event variables to be included in the OUTEVENT= data set.

Also note that even though the OUTEVENT= data set can not contain any periodic time series variables, the OUT= data set can contain event variables if they are explicitly specified in a KEEP statement. In other words, you can specify event variables in a KEEP statement, but you cannot specify periodic time series variables in a KEEPEVENT statement.

While variable selection for OUT= and OUTEVENT= data sets are controlled by a different set of statements (KEEP versus KEEPEVENT or DROP versus DROPEVENT), cross-section and range selections are controlled by the same statements. In other words, the WHERE and the RANGE statements are effective for both output data sets.

Supported File Types

PROC DATASOURCE can process only certain kinds of data files. For certain time series databases, the DATASOURCE procedure has built-in information on the layout of files comprising the database. PROC DATASOURCE knows how to read only these kinds of data files. To access these databases, you must indicate the data file type in the FILETYPE= option. For more detailed information, see the corresponding document for each filetype. See the section "References" on page 729.

The currently supported file types are summarized in Table 14.3.

Table 14.3. Supported File Types

Supplier	FILETYPE=	Description
BEA	BEANIPA	National Income and Product Accounts Tape Format
	BEANIPAD	National Income and Product Accounts Diskette Format
BLS	BLSCPI	Consumer Price Index Surveys
	BLSWPI	Producer Price Index Survey
	BLSEENA	National Employment, Hours, and Earnings Survey
	BLSEESA	State and Area Employment Hours and Earnings Survey
DRI	DRIBASIC	Basic Economic (formerly CITIBASE) Data Files
	CITIBASE	Tape Format CITIBASE Data Files
	DRIDDS	DRI Data Delivery Service Time Series
	CITIDISK	PC Diskette format CITIBASE Databases
CRSP	CRY2DBS	Y2K Daily Binary Security File Format
	CRY2DBI	Y2K Daily Binary Calendar&Indices File Format
	CRY2DBA	Y2K Daily Binary File Annual Data Format
	CRY2MBS	Y2K Monthly Binary Security File Format
	CRY2MBI	Y2K Monthly Binary Calendar&Indices File Format
	CRY2MBA	Y2K Monthly Binary File Annual Data Format
CRSP	CRY2DCS	Y2K Daily Character Security File Format
	CRY2DCI	Y2K Daily Character Calendar&Indices File Format
	CRY2DCA	Y2K Daily Character File Annual Data Format
	CRY2MCS	Y2K Monthly Character Security File Format
	CRY2MCI	Y2K Monthly Character Calendar&Indices File Format
	CRY2MCA	Y2K Monthly Character File Annual Data Format
CRSP	CRY2DIS	Y2K Daily IBM Binary Security File Format
	CRY2DII	Y2K Daily IBM Binary Calendar&Indices File Format
	CRY2DIA	Y2K Daily IBM Binary File Annual Data Format
	CRY2MIS	Y2K Monthly IBM Binary Security File Format

Table 14.3. (continued)

Supplier	FILETYPE=	Description
	CRY2MII	Y2K Monthly IBM Binary Calendar&Indices File Format
	CRY2MIA	Y2K Monthly IBM Binary File Annual Data Format
CRSP	CRY2MVS	Y2K Monthly VAX Binary Security File Format
	CRY2MVI	Y2K Monthly VAX Binary Calendar&Indices File Format
	CRY2MVA	Y2K Monthly VAX Binary File Annual Data Format
	CRY2DVS	Y2K Daily VAX Binary Security File Format
	CRY2DVI	Y2K Daily VAX Binary Calendar&Indices File Format
	CRY2DVA	Y2K Daily VAX Binary File Annual Data Format
CRSP	CRSPDBS	CRSP Daily Binary Security File Format
	CRSPDBI	CRSP Daily Binary Calendar&Indices File Format
	CRSPDBA	CRSP Daily Binary File Annual Data Format
	CRSPMBS	CRSP Monthly Binary Security File Format
	CRSPMBI	CRSP Monthly Binary Calendar&Indices File Format
	CRSPMBA	CRSP Monthly Binary File Annual Data Format
CRSP	CRSPDCS	CRSP Daily Character Security File Format
	CRSPDCI	CRSP Daily Character Calendar&Indices File Format
	CRSPDCA	CRSP Daily Character File Annual Data Format
	CRSPMCS	CRSP Monthly Character Security File Format
	CRSPMCI	CRSP Monthly Character Calendar&Indices File Format
	CRSPMCA	CRSP Monthly Character File Annual Data Format
CRSP	CRSPDIS	CRSP Daily IBM Binary Security File Format
	CRSPDII	CRSP Daily IBM Binary Calendar&Indices File Format
	CRSPDIA	CRSP Daily IBM Binary File Annual Data Format
	CRSPMIS	CRSP Monthly IBM Binary Security File Format
	CRSPMII	CRSP Monthly IBM Binary Calendar&Indices File Format
	CRSPMIA	CRSP Monthly IBM Binary File Annual Data Format
CRSP	CRSPMVS	CRSP Monthly VAX Binary Security File Format
	CRSPMVI	CRSP Monthly VAX Binary Calendar&Indices File Format
	CRSPMVA	CRSP Monthly VAX Binary File Annual Data Format
	CRSPDVS	CRSP Daily VAX Binary Security File Format
	CRSPDVI	CRSP Daily VAX Binary Calendar&Indices File Format
	CRSPDVA	CRSP Daily VAX Binary File Annual Data Format
CRSP	CRSPMUS	CRSP Monthly UNIX Binary Security File Format
ACCESS97		CRSP ACCESS97 Monthly Security File Format
	CRSPMUI	CRSP Monthly UNIX Binary Calendar&Indices File Format
		CRSP ACCESS97 Monthly Calendar&Indices File Format
	CRSPMUA	CRSP Monthly UNIX Binary File Annual Data Format
		CRSP ACCESS97 Monthly Annual Data File Format
CRSP	CRSPDUS	CRSP Daily UNIX Binary Security File Format
ACCESS97		CRSP ACCESS97 Daily Security File Format
	CRSPDUI	CRSP Daily UNIX Binary Calendar&Indices File Format

Table 14.3. (continued)

Supplier	FILETYPE=	Description
		CRSP ACCESS97 Daily Calendar&Indices File Format
	CRSPDUA	CRSP Daily UNIX Binary File Annual Data Format
		CRSP ACCESS97 Daily Annual Data File Format
CRSP	CRSPMOS	CRSP Monthly Old Character Security File Format
	CRSPMOI	CRSP Monthly Old Character Calendar&Indices File Format
	CRSPMOA	CRSP Monthly Old Character File Annual Data Format
	CRSPDOS	CRSP Daily Old Character Security File Format
	CRSPDOI	CRSP Daily Old Character Calendar&Indices File Format
	CRSPDOA	CRSP Daily Old Character File Annual Data Format
CRSP	CR95MIS	CRSP 1995 Monthly IBM Binary Security File Format
	CR95MII	CRSP 1995 Monthly IBM Binary Calendar&Indices File Format
	CR95MIA	CRSP 1995 Monthly IBM Binary File Annual Data Format
	CR95DIS	CRSP 1995 Daily IBM Binary Security File Format
	CR95DII	CRSP 1995 Daily IBM Binary Calendar&Indices File Format
	CR95DIA	CRSP 1995 Daily IBM Binary File Annual Data Format
CRSP	CR95MVS	CRSP 1995 Monthly VAX Binary Security File Format
	CR95MVI	CRSP 1995 Monthly VAX Binary Calendar&Indices File Format
	CR95MVA	CRSP 1995 Monthly VAX Binary File Annual Data Format
	CR95DVS	CRSP 1995 Daily VAX Binary Security File Format
	CR95DVI	CRSP 1995 Daily VAX Binary Calendar&Indices File Format
	CR95DVA	CRSP 1995 Daily VAX Binary File Annual Data Format
CRSP	CR95MUS	CRSP 1995 Monthly UNIX Binary Security File Format
	CR95MUI	CRSP 1995 Monthly UNIX Binary Calendar&Indices File Format
	CR95MUA	CRSP 1995 Monthly UNIX Binary File Annual Data Format
	CR95DUS	CRSP 1995 Daily UNIX Binary Security File Format
	CR95DUI	CRSP 1995 Daily UNIX Binary Calendar&Indices File Format
	CR95DUA	CRSP 1995 Daily UNIX Binary File Annual Data Format
CRSP	CR95MSS	CRSP 1995 Monthly VMS Binary Security File Format
	CR95MSI	CRSP 1995 Monthly VMS Binary Calendar&Indices File Format
	CR95MSA	CRSP 1995 Monthly VMS Binary File Annual Data Format
	CR95DSS	CRSP 1995 Daily VMS Binary Security File Format
	CR95DSI	CRSP 1995 Daily VMS Binary Calendar&Indices File Format

Table 14.3. (continued)

Supplier	FILETYPE=	Description
	CR95DSA	CRSP 1995 Daily VMS Binary File Annual Data Format
CRSP	CR95MAS	CRSP 1995 Monthly ALPHA Binary Security File Format
	CR95MAI	CRSP 1995 Monthly ALPHA Binary Calendar&Indices File Format
	CR95MAA	CRSP 1995 Monthly ALPHA Binary File Annual Data Format
	CR95DAS	CRSP 1995 Daily ALPHA Binary Security File Format
	CR95DAI	CRSP 1995 Daily ALPHA Binary Calendar&Indices File Format
	CR95DAA	CRSP 1995 Daily ALPHA Binary File Annual Data Format
Haver	HAVER	Haver Analytics Data Files
IMF	IMFIFSP	International Financial Statistics, Packed Format
	IMFDOTSP	Direction of Trade Statistics, Packed Format
	IMFBOPSP	Balance of Payment Statistics, Packed Format
	IMFGFSP	Government Finance Statistics, Packed Format
OECD	OECDANA	OECD Annual National Accounts Tape Format
	OECDQNA	OECD Quarterly National Accounts Tape Format
	OECDMEI	OECD Main Economic Indicators Tape Format
S&P	CSAIBM	COMPUSTAT Annual, IBM 360&370 Format
	CS48QIBM	COMPUSTAT 48 Quarter, IBM 360&370 Format
	CSAUC	COMPUSTAT Annual, Universal Character Format
	CS48QUC	COMPUSTAT 48 Quarter, Universal Character Format
S&P	CSAIY2	Y2K COMPUSTAT Annual, IBM 360&370 Format
	CSQIY2	Y2K COMPUSTAT 48 Quarter, IBM 360&370 Format
	CSAUCY2	Y2K COMPUSTAT Annual, Universal Character Format
	CSQUCY2	Y2K COMPUSTAT 48 Quarter, Universal Character Format

Data supplier abbreviations used in Table 14.3 are

Abbreviation	Supplier
BEA	Bureau of Economic Analysis, U.S. Department of Commerce
BLS	Bureau of Labor Statistics, U.S. Department of Labor
CRSP	Center for Research in Security Prices
DRI	DRIMcGraw-Hill
FAME	FAME Information Services, Inc
Haver	Haver Analytics Inc.
IMF	International Monetary Fund
OECD	Organization for Economic Cooperation and Development
S&P	Standard & Poor's Compustat Services Inc.

BEA Data Files

The Bureau of Economic Analysis, U.S. Department of Commerce, supplies national income, product accounting, and various other macro economic data at the regional, national, and international levels in the form of data files with various formats and on various media.

The following BEA data file types are supported:

FILETYPE=BEANIPA–National Income and Product Accounts Tape Format

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	PARTNO	Part Number of Publication, Integer Portion of the Table Number, 1-9 (character)
	TABNUM	Table Number Within Part, Decimal Portion of the Table Number, 1-24 (character)
Series Variables	Series variable names are constructed by concatenating table number suffix, line and column numbers within each table. An underscore (_) prefix is also added for readability.	

FILETYPE=BEANIPAD–National Income and Product Accounts Diskette Format

The diskette format National Income and Product Accounts files contain the same information as the tape format files described previously.

Data Files	Database is stored in a single diskette file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	PARTNO	Part Number of Publication, Integer Portion of the Table Number, 1-9 (character)
	TABNUM	Table Number Within Part, Decimal Portion of the Table Number, 1-24 (character)
Series Variables	Series variable names are constructed by concatenating table number suffix, line and column numbers within each table. An underscore (_) prefix is also added for readability.	

BLS Data Files

The Bureau of Labor Statistics, U.S. Department of Labor, compiles and distributes data on employment, expenditures, prices, productivity, injuries and illnesses, and wages.

The following BLS file types are supported:

FILETYPE=BLSCPI–Consumer Price Index Surveys (=CU,CW)

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, SEMIYEAR1.6, MONTH (default)	
BY variables	SURVEY	Survey type: CU=All Urban Consumers, CW=Urban Wage Earners and Clerical Workers (character)
	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	AREA	Geographic Area (character)
	BASPTYPE	Index Base Period Type, S=Standard, A=Alternate Reference (character)
	BASEPER	Index Base Period (character)
Series Variables	Series variable names are the same as consumer item codes listed in the Series Directory shipped with the data tapes.	
Missing Codes	A data value of 0 is interpreted as MISSING.	

FILETYPE=BLSWPI–Producer Price Index Survey (WP)

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, MONTH (default)	
BY variables	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	MAJORCOM	Major Commodity Group (character)
Sorting Order	BY SEASON MAJORCOM	
Series Variables	Series variable names are the same as commodity codes but prefixed by an underscore (_).	
Missing Codes	A data value of 0 is interpreted as MISSING.	

FILETYPE=BLSEENA–National Employment, Hours, and Earnings Survey

Data Files	Database is stored in a single tape file.
INTERVAL=	YEAR, QUARTER, MONTH (default)

BY variables	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	DIVISION	Major Industrial Division (character)
	INDUSTRY	Industry Code (character)
Sorting Order	BY SEASON DIVISION INDUSTRY	
Series Variables	Series variable names are the same as data type codes prefixed by EE.	
	EE01	Total Employment,
	EE02	Employment of Women
	EE03	Employment of Production or Nonsupervisory Workers
	EE04	Average Weekly Earnings of Production Workers
	EE05	Average Weekly Hours of Production Workers
	EE06	Average Hourly Earnings of Production Workers
	EE07	Average Weekly Overtime Hours of Production Workers
	EE40	Index of Aggregate Weekly Hours
	EE41	Index of Aggregate Weekly Payrolls
	EE47	Hourly Earnings Index; 1977 Weights; Current Dollars
	EE48	Hourly Earnings Index; 1977 Weights; Base 1977 Dollars
	EE49	Average Hourly Earnings; Base 1977 Dollars
	EE50	Gross Average Weekly Earnings; Current Dollars
	EE51	Gross Average Weekly Earnings; Base 1977 Dollars
	EE52	Spendable Average Weekly Earnings; No Dependents; Current Dollars
	EE53	Spendable Average Weekly Earnings; No Dependents; Base 1977 Dollars
	EE54	Spendable Average Weekly Earnings; 3 Dependents; Current Dollars
	EE55	Spendable Average Weekly Earnings; 3 Dependents; Base 1977 Dollars
	EE60	Average Hourly Earnings Excluding Overtime
	EE61	Index of Diffusion; 1-month Span; Base 1977
	EE62	Index of Diffusion; 3-month Span; Base 1977
	EE63	Index of Diffusion; 6-month Span; Base 1977
	EE64	Index of Diffusion; 12-month Span; Base 1977
Missing Codes	Series data values are set to MISSING when their status codes are 1.	

FILETYPE=BLSEESA–State and Area Employment, Hours, and Earnings Survey

Data Files	Database is stored in a single tape file.	
INTERVAL=	YEAR, MONTH (default)	
BY variables	STATE	State FIPS codes (numeric)
	AREA	Area Codes (character)
	DIVISION	Major Industrial Division (character)
	INDUSTRY	Industry Code (character)
	DETAIL	Private/Government Detail
Sorting Order	BY STATE AREA DIVISION INDUSTRY DETAIL	
Series Variables	Series variable names are the same as data type codes prefixed by SA.	
	SA1	All employees
	SA2	Women workers
	SA3	Production Workers
	SA4	Average weekly earnings
	SA5	Average weekly hours
Missing Codes	Series data values are set to MISSING when their status codes are 1.	

DRI/McGraw-Hill Data Files

The DRIBASIC (formerly CITIBASE) database contains economic and financial indicators of the U.S. and international economies gathered from various government and private sources by DRI/McGraw-Hill, Inc. There are over 8000 yearly, quarterly, monthly, weekly, and daily time series.

DRI/McGraw-Hill distributes Basic Economic data files on various media. DRI also offers Data Delivery Service (DDS)data files via DRIPRO’s data retrieval software called Xtract. Most DDS data files can be read by DATASOURCE using the DRIDDS filetype.

The following DRI file types are supported:

FILETYPE=DRIBASIC–DRI Basic Economic Data Files

Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), QUARTER, MONTH, WEEK, WEEK1.1, WEEK1.2, WEEK1.3, WEEK1.4, WEEK1.5, WEEK1.6, WEEK1.7, WEEKDAY
BY variables	None
Series Variables	Variable names are taken from the series descriptor records in the data file . Note that series codes can be 20 bytes.
Missing Codes	MISSING=(’1.000000E9’=. ’NA’-’ND’=.)

Note that when you specify the INTERVAL=WEEK option, all the weekly series will be aggregated, and the DATE variable in the OUT= data set will be set to the date of Sundays. The date of first observation for each series is the Sunday marking the beginning of the week that contains the starting date of that variable.

FILETYPE=DRIDDS–DRI Data Delivery Service Data Files

Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), SEMIYEAR, QUARTER, MONTH, SEMIMONTH, TENDAY, WEEK, WEEK1.1, WEEK1.2, WEEK1.3, WEEK1.4, WEEK1.5, WEEK1.6, WEEK1.7, WEEKDAY, DAY
BY variables	None
Series Variables	Variable names are taken from the series descriptor records in the data file . Note that series names can be 24 bytes.
Missing Codes	MISSING=(’NA’-’ND’=.)

FILETYPE=CITIOLD–Old format CITIBASE data files

This file type is used for CITIBASE data tapes distributed prior to May, 1987.

Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), QUARTER, MONTH
BY variables	None
Series Variables	Variable names are taken from the series descriptor records in the data file and are the same as the series codes reported in the *CITIBASE Directory*.
Missing Codes	1.0E9=.

FILETYPE=CITIDISK–PC Diskette Format CITIBASE Databases

Data Files	Database is stored in groups of three associated files having the same file name but different extensions: KEY, IND, or DB. The INFILE= option should contain three filerefs in the following order: INFILE=(*keyfile indfile dbfile*)
INTERVAL=	YEAR (default), QUARTER, MONTH
BY variables	None
Series Variables	Series variable names are the same as series codes reported in the *CITIBASE Directory*.
Missing Codes	1.0E9=.

COMPUSTAT Data Files

COMPUSTAT data files, distributed by Standard and Poor's Compustat Services, Inc., consist of a collection of financial, statistical, and market information covering several thousand industrial and nonindustrial companies. Data are available in both an IBM 360/370 format and a "Universal Character" format, both of which further subdivide into annual and quarterly formats.

The BY variables are used to select individual companies or a group of companies. Individual companies can be selected by their unique six-digit CUSIP issuer code (CNUM). A number of specific groups of companies can be extracted from the tape by the following key fields:

FILE	specifies the file identification code used to group companies by files
ZLIST	specifies the exchange listing code that can be used to group companies by exchange
DNUM	is used to extract companies in a specific SIC industry group

Series names are internally constructed from the data array names documented in the COMPUSTAT manual. Each column of data array is treated as a SAS variable. The names of these variables are generated by concatenating the corresponding column numbers to the array name.

Missing values use four codes. Missing code '.C' represents a combined figure where the data item has been combined into another data item, '.I' reports an insignificant figure, '.S' represents a semi-annual figure in the second and fourth quarters, '.A' represents an annual figure in the fourth quarter, and '.' indicates that the data item is not available. The missing codes '.C' and '.I' are not used for Aggregate or Prices, Dividends, and Earnings (PDE) files. The missing codes '.S' and '.A' are used only on the Industrial Quarterly File and not on the Aggregate Quarterly, Business Information, or PDE files.

FILETYPE=CSAIBM–COMPUSTAT Annual, IBM 360/370 Format

FILETYPE=CSAIY2–Four-Digit Year COMPUSTAT Annual, IBM 360/370 Format

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default)	
BY variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (numeric)
	FILE	File Identification Code (numeric)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)

	SMBL	Stock Ticker Symbol (character)
	XREL	S&P Industry Index Relative Code (numeric)
	STK	Stock Ownership Code (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	FINC	Incorporation Code - Foreign (numeric)
	EIN	Employer Identification Number (character)
	CPSPIN	S&P Index Primary Marker (character)
	CSSPIN	S&P Index Secondary Identifier (character)
	CSSPII	S&P Index Subset Identifier (character)
	SDBT	S&P Senior Debt Rating - Current (character)
	SDBTIM	Footnote- S&P Senior Debt Rating- Current (character)
	SUBDBT	S&P Subordinated Debt Rating - Current (character)
	CPAPER	S&P Commercial Paper Rating - Current (character)
Sorting order	BY DNUM CNUM CIC	
Series Variables	DATA1-DATA350 FYR UCODE SOURCE AFTNT1-AFTNT70	
Default KEEP List	DROP DATA322-DATA326 DATA338 DATA345-DATA347 DATA350 AFTNT52-AFTNT70;	
Missing Codes	0.0001=. 0.0004=.C 0.0008=.I 0.0002=.S 0.0003=.A	

FILETYPE=CS48QIBM–COMPUSTAT 48-Quarter, IBM 360/370 Format

FILETYPE=CSQIY2–FOUR-DIGIT YEAR COMPUSTAT 48-Quarter, IBM 360/370 Format

Data Files	Database is stored in a single file.	
INTERVAL=	QUARTER (default)	
BY variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (numeric)
	FILE	File Identification Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	EIN	Employer Identification Number (character)
	STK	Stock Ownership Code (numeric)
	SMBL	Stock Ticker Symbol (character)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	XREL	S&P Industry Index Relative Code (numeric)
	FIC	Incorporation Code - Foreign (numeric)

	INCORP	Incorporation Code - State (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	CANDX	Canadian Index Code - Current (character)
Sorting order	BY DNUM CNUM CIC;	
Series Variables	DATA1-DATA232	Data Array
	QFTNT1-QFTNT60	Data Footnotes
	FYR	Fiscal Year-end Month of Data
	SPCSCYR	SPCS Calendar Year
	SPCSCQTR	SPCS Calendar Quarter
	UCODE	Update Code
	SOURCE	Source Document Code
	BONDRATE	S&P Bond Rating
	DEBTCL	S&P Class of Debt
	CPRATE	S&P Commercial Paper Rating
	STOCK	S&P Common Stock Ranking
	MIC	S&P Major Index Code
	IIC	S&P Industry Index Code
	REPORTDT	Report Date of Quarterly Earnings
	FORMAT	Flow of Funds Statement Format Code
	DEBTRT	S&P Subordinated Debt Rating
	CANIC	Canadian Index Code
	CS	Comparability Status
	CSA	Company Status Alert
	SENIOR	S&P Senior Debt Rating
Default KEEP List	DROP DATA122-DATA232 QFTNT24-QFTNT60;	
Missing Codes	0.0001=. 0.0004=.C 0.0008=.I 0.0002=.S 0.0003=.A	

FILETYPE=CSAUC–COMPUSTAT Annual, Universal Character Format

FILETYPE=CSAUCY2–Four-Digit Year COMPUSTAT Annual, Universal Character Format

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default)	
BY variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (character)
	FILE	File Identification Code (numeric)

	ZLIST	Exchange Listing and S&P Index Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	SMBL	Stock Ticker Symbol (character)
	XREL	S&P Industry Index Relative Code (numeric)
	STK	Stock Ownership Code (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	FINC	Incorporation Code - Foreign (numeric)
	EIN	Employer Identification Number (character)
	CPSPIN	S&P Index Primary Marker (character)
	CSSPIN	S&P Index Secondary Identifier (character)
	CSSPII	S&P Index Subset Identifier (character)
	SDBT	S&P Senior Debt Rating - Current (character)
	SDBTIM	Footnote- S&P Senior Debt Rating- Current (character)
	SUBDBT	S&P Subordinated Debt Rating - Current (character)
	CPAPER	S&P Commercial Paper Rating - Current (character)
Sorting order	BY DNUM CNUM CIC	
Series Variables	DATA1-DATA350 FYR UCODE SOURCE AFTNT1-AFTNT70	
Default KEEP List	DROP DATA322-DATA326 DATA338 DATA345-DATA347 DATA350 AFTNT52-AFTNT70;	
Missing Codes	-0.001=. -0.004=.C -0.008=.I -0.002=.S -0.003=.A	

FILETYPE=CS48QUC–COMPUSTAT 48 Quarter, Universal Character Format

FILETYPE=CSQUCY2–Four-Digit Year COMPUSTAT 48 Quarter, Universal Character Format

Data Files	Database is stored in a single file.	
INTERVAL=	QUARTER (default)	
BY variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (character)
	FILE	File Identification Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	EIN	Employer Identification Number (character)
	STK	Stock Ownership Code (numeric)
	SMBL	Stock Ticker Symbol (character)

	ZLIST	Exchange Listing and S&P Index Code (numeric)
	XREL	S&P Industry Index Relative Code (numeric)
	FIC	Incorporation Code - Foreign (numeric)
	INCORP	Incorporation Code - State (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	CANDXC	Canadian Index Code - Current (numeric)
Sorting order	BY DNUM CNUM CIC	
Series Variables	DATA1-DATA232	Data Array
	QFTNT1-QFTNT60	Data Footnotes
	FYR	Fiscal Year-end Month of Data
	SPCSCYR	SPCS Calendar Year
	SPCSCQTR	SPCS Calendar Quarter
	UCODE	Update Code
	SOURCE	Source Document Code
	BONDRATE	S&P Bond Rating
	DEBTCL	S&P Class of Debt
	CPRATE	S&P Commercial Paper Rating
	STOCK	S&P Common Stock Ranking
	MIC	S&P Major Index Code
	IIC	S&P Industry Index Code
	REPORTDT	Report Date of Quarterly Earnings
	FORMAT	Flow of Funds Statement Format Code
	DEBTRT	S&P Subordinated Debt Rating
	CANIC	Canadian Index Code - Current
	CS	Comparability Status
	CSA	Company Status Alert
	SENIOR	S&P Senior Debt Rating
Default KEEP List	DROP DATA122-DATA232 QFTNT24-QFTNT60;	
Missing Codes	-0.001=. -0.004=.C -0.008=.I -0.002=.S -0.003=.A	

CRSP Stock Files

The Center for Research in Security Prices provides comprehensive security price data through two primary stock files, the NYSE/AMEX file and the NASDAQ file. These files are composed of master and return components, available separately or combined. CRSP stock files are further differentiated by the frequency at which prices and returns are reported, daily or monthly. Both daily and monthly files contain annual data fields.

CRSP data files come either in binary or character tape format, or in CRSPAccess CDROM format. See Chapter 5, “The SASECRSP Interface Engine,” for more about accessing your CRSPAccess database. Use the Datasource procedure for serial access of the tape file formats.

CRSP stock data are provided in two files, a main data file containing security information and a calendar/indices file containing a list of trading dates and market information associated with those trading dates. If security data do not fit on one tape, they are split into two or more files, each one of which resides on a different self-contained tape. The calendar/indices file is on the first tape only.

The file types for CRSP stock files are constructed by concatenating CRSP with a D or M to indicate the frequency of data, followed by B,C, or I to indicate file formats. B is for host binary, C is for character, and I is for IBM binary formats. The last character in the file type indicates if you are reading the Calendar/Indices file (I), or if you are extracting the security (S) or annual data (A). For example, the file type for the daily NYSE/AMEX combined tape in IBM binary format is CRSPDIS. Its calendar/indices file can be read by CRSPDII, and its annual data can be extracted by CRSPDIA.

Starting in 1995, binary data tapes use split records (RICFAC=2) so the 1995 filetypes (CR95*) should be used for 1995 and 1996 binary data.

If you use utility routines supplied by CRSP to convert a character format file to a binary format file on a given host, then you need to use host binary file types (RIDFAC=1) to read those files in. Note that you can not do the conversion on one host and transfer and read the file on another host.

If you are using the CRSP Access97 Database, you will need to use the utility routine (stk_dump_bin) supplied by CRSP to generate the UNIX binary format of the data. You can access the UNIX (or SUN) binary data by using PROC DATASOURCE with the CRSPDUS for daily or CRSPMUS for monthly stock data. See the example on Example 14.11 later in this chapter.

For the four-digit year data, use the Y2K compliant filetypes for that data type.

For CRSP file types, the INFILE= option must be of the form

```
   INFILE=( calfile security1 < security2 ... > )
```

where *calfile* is the fileref assigned to the calendar/indices file, and *securty1* < *securty2* ... > are the filerefs given to the security files, in the order in which they should be read.

CRSP Calendar/Indices Files

Data Files	Database is stored in a single file.	
INTERVAL=	DAY	for products DA, DR, DX, EX, NX and RA
	MONTH	for products MA, MX and MZ

BY variables	None	
Series Variables	VWRETD	Value-Weighted Return (including all distributions)
	VWRETX	Value-Weighted Return (excluding dividends)
	EWRETD	Equal-Weighted Return (including all distributions)
	EWRETX	Equal-Weighted Return (excluding dividends)
	TOTVAL	Total Market Value
	TOTCNT	Total Market Count
	USDVAL	Market Value of Securities Used
	USDCNT	Count of Securities Used
	SPINDX	Level of the Standard & Poor's Composite Index
	SPRTRN	Return on the Standard & Poor's Composite Index
	NCINDX	NASDAQ Composite Index
	NCRTRN	NASDAQ Composite Return
Default KEEP List	All variables will be kept.	

CRSP Daily Security Files

Data Files	INFILE=(calfile securty1 < securty2 ... >)		
INTERVAL=	DAY		
BY variables	CUSIP	CUSIP Identifier (character)	
	PERMNO	CRSP Permanent Number (numeric)	
	COMPNO	NASDAQ Company Number (numeric)	
	ISSUNO	NASDAQ Issue Number (numeric)	
	HEXCD	Header Exchange Code (numeric)	
	HSICCD	Header SIC Code (numeric)	
Sorting Order	BY CUSIP		
Series Variables	BIDLO	Bid or Low	
	ASKHI	Ask or High	
	PRC	Closing Price of Bid/Ask Average	
	VOL	Share Volume	
	RET	Holding Period Return missing=(-66.0 = .p -77.0 = .t -88.0 = .r -99.0 = .b)	
	BXRET	Beta Excess Return missing=(-44.0 = .)	
	SXRET	Standard Deviation Excess Return missing=(-44.0 = .)	
Events	NAMES	NCUSIP	Name CUSIP
		TICKER	Exchange Ticker Symbol
		COMNAM	Company Name
		SHRCLS	Share Class

		SHRCD	Share Code
		EXCHCD	Exchange Code
		SICCD	Standard Industrial Classification Code
	DIST	DISTCD	Distribution Code
		DIVAMT	Dividend Cash Amount
		FACPR	Factor to Adjust Price
		FACSHR	Factor to Adjust Shares Outstanding
		DCLRDT	Declaration Date
		RCRDDT	Record Date
		PAYDT	Payment Date
	SHARES	SHROUT	Number of Shares Outstanding
		SHRFLG	Share Flag
	DELIST	DLSTCD	Delisting Code
		NWPERM	New CRSP Permanent Number
		NEXTDT	Date of Next Available Information
		DLBID	Delisting Bid
		DLASK	Delisting Ask
		DLPRC	Delisting Price
		DLVOL	Delisting Volume missing=(-99 = .)
		DLRET	Delisting Return missing=(-55.0=.s -66.0=.t -88.0=.a -99.0=.p);
	NASDIN	TRTSCD	Traits Code
		NMSIND	National Market System Indicator
		MMCNT	Market Maker Count
		NSDINX	NASD Index
Default KEEP Lists	All periodic series variables will be output to the OUT= data set and all event variables will be output to the OUTEVENT= data set.		

CRSP Monthly Security Files

Data Files	INFILE=(calfile securty1 < securty2 ... >)	
INTERVAL=	MONTH	
BY variables	CUSIP	CUSIP Identifier (character)
	PERMNO	CRSP Permanent Number (numeric)
	COMPNO	NASDAQ Company Number (numeric)
	ISSUNO	NASDAQ Issue Number (numeric)
	HEXCD	Header Exchange Code (numeric)

	HSICCD	Header SIC Code (numeric)	
Sorting Order	BY CUSIP		
Series Variables	BIDLO	Bid or Low	
	ASKHI	Ask or High	
	PRC	Closing Price of Bid/Ask average	
	VOL	Share Volume	
	RET	Holding Period Return missing=(-66.0 = .p -77.0 = .t -88.0 = .r -99.0 = .b);	
	RETX	Return Without Dividends missing=(-44.0 = .)	
	PRC2	Secondary Price missing=(-44.0 = .)	
Events	NAMES	NCUSIP	Name CUSIP
		TICKER	Exchange Ticker Symbol
		COMNAM	Company Name
		SHRCLS	Share Class
		SHRCD	Share Code
		EXCHCD	Exchange Code
		SICCD	Standard Industrial Classification Code
	DIST	DISTCD	Distribution Code
		DIVAMT	Dividend Cash Amount
		FACPR	Factor to Adjust Price
		FACSHR	Factor to Adjust Shares Outstanding
		EXDT	Ex-distribution Date
		RCRDDT	Record Date
		PAYDT	Payment Date
	SHARES	SHROUT	Number of Shares Outstanding
		SHRFLG	Share Flag
	DELIST	DLSTCD	Delisting Code
		NWPERM	New CRSP Permanent Number
		NEXTDT	Date of Next Available Information
		DLBID	Delisting Bid
		DLASK	Delisting Ask
		DLPRC	Delisting Price
		DLVOL	Delisting Volume
		DLRET	Delisting Return missing=(-55.0=.s -66.0=.t -88.0=.a -99.0=.p);
	NASDIN	TRTSCD	Traits Code
		NMSIND	National Market System Indicator
		MMCNT	Market Maker Count

	NSDINX	NASD Index
Default KEEP Lists	All periodic series variables will be output to the OUT= data set and all event variables will be output to the OUTEVENT= data set.	

CRSP Annual Data

Data Files	INFILE=(securty1 < securty2 ... >)	
INTERVAL=	YEAR	
BY variables	CUSIP	CUSIP Identifier (character)
	PERMNO	CRSP Permanent Number (numeric)
	COMPNO	NASDAQ Company Number (numeric)
	ISSUNO	NASDAQ Issue Number (numeric)
	HEXCD	Header Exchange Code (numeric)
	HSICCD	Header SIC Code (numeric)
Sorting Order	BY CUSIP	
Series Variables	CAPV	Year End Capitalization
	SDEVV	Annual Standard Deviation missing=(-99.0 = .)
	BETAV	Annual Beta missing=(-99.0 = .)
	CAPN	Year End Capitalization Portfolio Assignment
	SDEVN	Standard Deviation Portfolio Assignment
	BETAN	Beta Portfolio Assignment
Default KEEP Lists	All variables will be kept.	

FAME Information Services Databases

The DATASOURCE procedure provides access to FAME Information Services databases for UNIX-based systems only. For a more flexible FAME database access use the SASEFAME interface engine in Chapter 6, "The SASEFAME Interface Engine," which is supported for Release 8.2 on Windows, Solaris2, AIX, and HP-UX hosts. SASEFAME for Version 9 supports Windows, Solaris 8, AIX, LINUX, and DEC/OSF Digital UNIX.

The DATASOURCE interface to FAME requires a component supplied by FAME Information Services, Inc. Once this FAME component is installed on your system, you can use the DATASOURCE procedure to extract data from your FAME databases as follows:

- Specify FILETYPE=FAME on the PROC DATASOURCE statement.
- Specify the FAME database to access with a DBNAME='*fame-database*' option on the PROC DATASOURCE statement. The character string you specify

on the DBNAME= option is passed through to FAME; specify the value of this option as you would in accessing the database from within FAME software.

- Specify the output SAS data set to be created, the frequency of the series to be extracted, and other usual DATASOURCE procedure options as appropriate.
- Specify the time range to extract with a RANGE statement. The RANGE statement is required when extracting series from FAME databases.
- Specify the FAME series to be extracted with a KEEP statement. The items on the KEEP statement are passed through to FAME software; therefore, you can use any valid FAME expression to specify the series to be extracted. Put in quotes any FAME series name or expression that is not a valid SAS name.
- Specify the SAS variable names you want to use for the extracted series on a RENAME statement. Give the FAME series name or expression (in quotes if needed) followed by an equal sign and the SAS name. The RENAME statement is not required; however, if the FAME series name is not a valid SAS variable name, the DATASOURCE procedure will construct a SAS name by translating and truncating the FAME series name. This process may not produce the desired name for the variable in the output SAS data set, so a rename statement could be used to produce a more appropriate variable name. The VALIDVARNAME=ANY option on your SAS options statement can be used to allow special characters in the SAS variable name.

For an alternative solution to PROC DATASOURCE's access to FAME, see "The SASEFAME Interface Engine" in Chapter 6, "The SASEFAME Interface Engine."

FILETYPE=FAME–FAME Information Services Databases

INTERVAL=	YEAR	corresponds to FAME's ANNUAL(DECEMBER)
	YEAR.2	correspond to FAME's ANNUAL(JANUARY)
	YEAR.3	correspond to FAME's ANNUAL(FEBRUARY)
	YEAR.4	correspond to FAME's ANNUAL(MARCH)
	YEAR.5	correspond to FAME's ANNUAL(APRIL)
	YEAR.6	correspond to FAME's ANNUAL(MAY)
	YEAR.7	correspond to FAME's ANNUAL(JUNE)
	YEAR.8	correspond to FAME's ANNUAL(JULY)
	YEAR.9	correspond to FAME's ANNUAL(AUGUST)
	YEAR.10	correspond to FAME's ANNUAL(SEPTEMBER)
	YEAR.11	correspond to FAME's ANNUAL(OCTOBER)
	YEAR.12	correspond to FAME's ANNUAL(NOVEMBER)
	SEMIYEAR, QUARTER, MONTH, SEMIMONTH, TENDAY	are supported frequencies
	WEEK	corresponds to FAME's WEEKLY(SATURDAY)

	WEEK.2	corresponds to FAME's WEEKLY(SUNDAY)
	WEEK.3	corresponds to FAME's WEEKLY(MONDAY)
	WEEK.4	corresponds to FAME's WEEKLY(TUESDAY)
	WEEK.5	corresponds to FAME's WEEKLY(WEDNESDAY)
	WEEK.6	corresponds to FAME's WEEKLY(THURSDAY)
	WEEK.7	corresponds to FAME's WEEKLY(FRIDAY)
	WEEK2	corresponds to FAME's BIWEEKLY(ASATURDAY)
	WEEK2.2	correspond to FAME's BIWEEKLY(ASUNDAY)
	WEEK2.3	correspond to FAME's BIWEEKLY(AMONDAY)
	WEEK2.4	correspond to FAME's BIWEEKLY(ATUESDAY)
	WEEK2.5	correspond to FAME's BIWEEKLY(AWEDNESDAY)
	WEEK2.6	correspond to FAME's BIWEEKLY(ATHURSDAY)
	WEEK2.7	correspond to FAME's BIWEEKLY(AFRIDAY)
	WEEK2.8	correspond to FAME's BIWEEKLY(BSATURDAY)
	WEEK2.9	correspond to FAME's BIWEEKLY(BSUNDAY)
	WEEK2.10	correspond to FAME's BIWEEKLY(BMONDAY)
	WEEK2.11	correspond to FAME's BIWEEKLY(BTUESDAY)
	WEEK2.12	correspond to FAME's BIWEEKLY(BWEDNESDAY)
	WEEK2.13	correspond to FAME's BIWEEKLY(BTHURSDAY)
	WEEK2.14	correspond to FAME's BIWEEKLY(BFRIDAY)
	WEEKDAY, DAY	are supported frequencies
BY variables	None	
Series Variables	Variable names are constructed from the FAME series codes. Note that series names are limited to 32 bytes.	

Haver Analytics Data Files

Haver Analytics offers a broad range of economic, financial, and industrial data for the U.S. and other countries. See "The SASEHAVR Interface Engine" in Chapter 7, "The SASEHAVR Interface Engine," for accessing your HAVER DLX database. SASEHAVR for Version 9 is experiemental and is supported on Windows only. Use the DATASOURCE procedure for serial access of the tape file formats. The tape format of Haver Analytics data files is similar to the CITIBASE format.

FILETYPE=HAVER–Haver Analytics Data Files HAVERO–Old format Haver Files

Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), QUARTER, MONTH
BY variables	1.0E9=.
Series Variables	Variable names are taken from the series descriptor records in the data file. NOTE: HAVER filetype reports the UPDATE and SOURCE in the OUTCONT= data set, while HAVERO does not.
Missing Codes	1.0E9=.

IMF Data Files

The International Monetary Fund's Economic Information System (EIS) offers tape subscriptions for their International Financial Statistics (IFS), Direction of Trade Statistics (DOTS), Balance of Payment Statistics (BOPS), and the Government Finance Statistics (GFS) databases. The first three contain annual, quarterly, and monthly data, while the GFS file has only annual data.

IMF data tapes are available for IBM mainframe systems (EBCDIC character coding) in both a packed and an unpacked format. PROC DATASOURCE supports only the packed format at this time.

FILETYPE=IMFIFSP–International Financial Statistics, Packed format

The IFS data files contain over 23,000 time series including interest and exchange rates, national income and product accounts, price and production indexes, money and banking, export commodity prices, and balance of payments for nearly 200 countries and regional aggregates.

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	COUNTRY	Country Code (character, three-digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three-digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by F for data and F_F for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFDOTSP–Direction of Trade Statistics, Packed Format

The DOTS files contain time series on the distribution of exports and imports for about 160 countries and country groups by partner country and areas.

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	COUNTRY	Country Code (character, three-digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three-digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by D for data and F_D for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFBOPSP–Balance of Payment Statistics, Packed Format

The BOPS data files contain approximately 43,000 time series on balance of payments for about 120 countries.

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	COUNTRY	Country Code (character, three-digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three-digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by B for data and F_B for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFGFSP–Government Finance Statistics, Packed Format

The GFS data files encompass approximately 28,000 time series that give a detailed picture of federal government revenue, grants, expenditures, lending minus repayment financing and debt, and summary data of state and local governments, covering 128 countries.

Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY variables	COUNTRY	Country Code (character, three-digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three-digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by G for data and F_G for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

OECD Data Files

The Organization for Economic Cooperation and Development compiles and distributes statistical data, including National Accounts and Main Economic Indicators.

FILETYPE=OECDANA–Annual National Accounts

The ANA data files contain both main national aggregates accounts (Volume I) and detailed tables for each OECD Member country (Volume II).

Data Files	Database is stored on a single tape file.	
INTERVAL=	YEAR (default), SEMIYR1.6, QUARTER, MONTH, WEEK, WEEKDAY	
BY variables	PREFIX	Table number prefix (character)
	CNTRYZ	Country Code (character)
Series Variables	Series variable names are the same as the mnemonic name of the element given on the element 'E' record. They are taken from the 12 byte time series 'T' record time series indicative.	

```
rename p0discgdpe=p0digdpe;
rename doll2gdpe=dol2gdpe;
rename doll3gdpe=dol3gdpe;
rename doll1gdpe=dol1gdpe;
rename ppp1gdpd=pp1gdpd;
rename ppp1gdpd1=pp1gdpd1;
rename p0itxgdpc=p0itgdpc;
rename p0itxgdps=p0itgdps;
rename p0subgdpc=p0sugdpc;
rename p0subgdps=p0sugdps;
rename p0cfcgdpc=p0cfgdpc;
rename p0cfcgdps=p0cfgdps;
rename p0discgdpc=p0dicgdc;
rename p0discgdps=p0dicgds;
```

Missing Codes	A data value of * is interpreted as MISSING.

FILETYPE=OECDQNA–Quarterly National Accounts

The QNA file contains the main aggregates of quarterly national accounts for 16 OECD Member Countries and on a selected number of aggregates for 4 groups of member countries: OECD-Total, OECD-Europe, EEC, and the 7 major countries.

Data Files	Database is stored on a single file.	
INTERVAL=	QUARTER(default),YEAR	
BY variables	COUNTRY	Country Code (character)
	SEASON	Seasonality S=Seasonally adjusted 0=raw data, not seasonally adjusted
	PRICETAG	Prices C=data at current prices R,L,M=data at constant prices P,K,J,V=implicit price index or volume index
Series Variables	Subject code used to distinguish series within countries. Series variables are prefixed by _ for data, C for control codes, and D for relative date.	
Default DROP List	By default all the control codes and relative dates will be dropped.	
Missing Codes	A data value of + or - is interpreted as MISSING.	

FILETYPE=OECDMEI–Main Economic Indicators

The MEI file contains all series found in Parts 1 and 2 of the publication *Main Economic Indicators*.

Data Files	Database is stored on a single file.	
INTERVAL=	YEAR(default),QUARTER,MONTH	
BY variables	COUNTRY	Country Code (character)
	CURRENCY	Unit of expression of the series.
	ADJUST	Adjustment 0,H,S,A,L=no adjustment 1,I=calendar or working day adjusted 2,B,J,M=seasonally adjusted by National Authorities 3,K,D=seasonally adjusted by OECD
Series Variables	Series variables are prefixed by _ for data, C for control codes, and D for relative date in weeks since last updated.	
Default DROP List	By default, all the control codes and relative dates will be dropped.	
Missing Codes	A data value of + or - is interpreted as MISSING.	

Examples

Example 14.1. BEA National Income and Product Accounts

In this example, exports and imports of goods and services are extracted to demonstrate how to work with a National Income and Product Accounts Tape file.

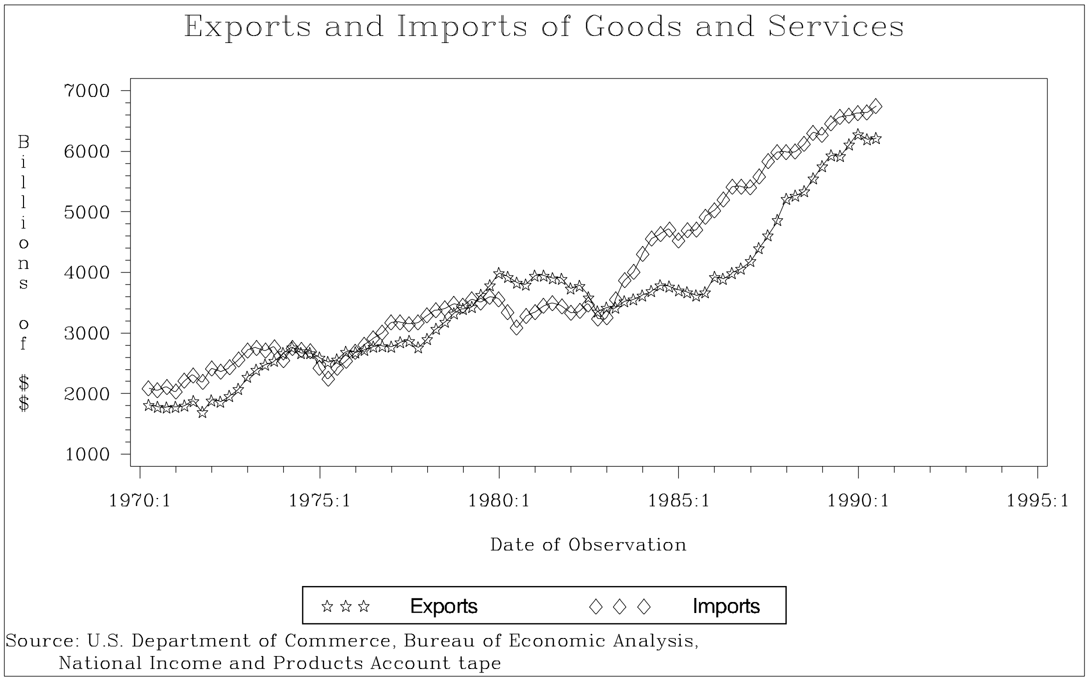
From the "Statistical Tables" published by the United States Department of Commerce, Bureau of Economic Analysis, exports and imports of goods and services are given in the second table (TABNUM='02') of the "Foreign Transactions" section (PARTNO='4'). This table does not have any table suffix A or B. Moreover, the first line in the table gives exports, while the eighth gives imports. Therefore, the series names for exports and imports are __00100 and __00800, where the first underscore is inserted by the procedure, the second underscore is the place holder for the table suffix, the following three-digits are the line numbers, and the last two-digits are the column numbers.

The following statements put this information together to extract quarterly exports and imports from a BEANIPA type file:

```
filename  datafile 'host-specific-path-name' host-options;
proc datasource filetype=beanipa infile=datafile
                interval=qtr out=foreign;
   keep __00100 __00800;
   where partno='4' and tabnum='02';
   label __00100='Exports of Goods and Services';
   label __00800='Imports of Goods and Services';
   rename __00100=exports __00800=imports;
run;
```

The plot of EXPORTS and IMPORTS against DATE is shown in Output 14.1.1.

Output 14.1.1. Plot of Time Series in the OUT= Data Set for FILETYPE=BEANIPA



This example illustrates the following features:

- You need to know the series variables names used by a particular vendor in order to construct the KEEP statement.
- You need to know the BY variable names and their values for the required cross sections.
- You can use RENAME and LABEL statements to associate more meaningful names and labels with your selected series variables.

Example 14.2. BLS Consumer Price Index Surveys

This example compares changes of the prices in medical care services with respect to different regions for all urban consumers (SURVEY='CU') since May, 1975. The source of data is the Consumer Price Index Surveys distributed by the U.S. Department of Labor, Bureau of Labor Statistics.

An initial run of PROC DATASOURCE gives the descriptive information on different regions available (the OUTBY= data set), as well as the series variable name corresponding to medical care services (the OUTCONT= data set).

```
filename datafile 'host-specific-file-name' <host-options>;
proc datasource filetype=blscpi interval=month
                outby=cpikey outcont=cpicont;
   where survey='CU';
run;
```

```
title1 'Partial Listing of the OUTBY= Data Set';
proc print data=cpikey noobs;
   where upcase(areaname) in
         ('NORTHEAST','NORTH CENTRAL','SOUTH','WEST');
run;

title1 'Partial Listing of the OUTCONT= Data Set';
proc print data=cpicont noobs;
   where index( upcase(label), 'MEDICAL CARE' );
run;
```

The OUTBY= data set in Output 14.2.1 lists all cross sections available for the four geographical regions: Northeast (AREA='0100'), North Central (AREA='0200'), Southern (AREA='0300'), and Western (AREA='0400'). The OUTCONT= data set gives the variable names for medical care related series.

Output 14.2.1. Partial Listings of the OUTBY= and OUTCONT= Data Sets

```
                    Partial Listing of the OUTBY= Data Set

survey    season    area    basptype    baseper             st_date    end_date

  CU        U       0100       A        DECEMBER 1977=100    DEC1966    JUL1990
  CU        U       0100       S        1982-84=100          DEC1966    JUL1990
  CU        U       0100       S        DECEMBER 1982=100    DEC1982    JUL1990
  CU        U       0100       S        DECEMBER 1986=100    DEC1986    JUL1990
  CU        U       0200       A        DECEMBER 1977=100    DEC1966    JUL1990
  CU        U       0200       S        1982-84=100          DEC1966    JUL1990
  CU        U       0200       S        DECEMBER 1982=100    DEC1982    JUL1990
  CU        U       0200       S        DECEMBER 1986=100    DEC1986    JUL1990
  CU        U       0300       A        DECEMBER 1977=100    DEC1966    JUL1990
  CU        U       0300       S        1982-84=100          DEC1966    JUL1990
  CU        U       0300       S        DECEMBER 1982=100    DEC1982    JUL1990
  CU        U       0300       S        DECEMBER 1986=100    DEC1986    JUL1990
  CU        U       0400       A        DECEMBER 1977=100    DEC1966    JUL1990
  CU        U       0400       S        1982-84=100          DEC1966    JUL1990
  CU        U       0400       S        DECEMBER 1982=100    DEC1982    JUL1990
  CU        U       0400       S        DECEMBER 1986=100    DEC1986    JUL1990

ntime    nobs    nseries    nselect       surtitle        areaname

 284      284       1          1      ALL URBAN CONSUM    NORTHEAST
 284      284      90         90      ALL URBAN CONSUM    NORTHEAST
  92       92       7          7      ALL URBAN CONSUM    NORTHEAST
  44       44       1          1      ALL URBAN CONSUM    NORTHEAST
 284      284       1          1      ALL URBAN CONSUM    NORTH CENTRAL
 284      284      90         90      ALL URBAN CONSUM    NORTH CENTRAL
  92       92       7          7      ALL URBAN CONSUM    NORTH CENTRAL
  44       44       1          1      ALL URBAN CONSUM    NORTH CENTRAL
 284      284       1          1      ALL URBAN CONSUM    SOUTH
 284      284      90         90      ALL URBAN CONSUM    SOUTH
  92       92       7          7      ALL URBAN CONSUM    SOUTH
  44       44       1          1      ALL URBAN CONSUM    SOUTH
 284      284       1          1      ALL URBAN CONSUM    WEST
 284      284      90         90      ALL URBAN CONSUM    WEST
  92       92       7          7      ALL URBAN CONSUM    WEST
  44       44       1          1      ALL URBAN CONSUM    WEST
```

```
                  Partial Listing of the OUTCONT= Data Set

        s
        e                                                     f    f
        l         l    v                                f     o    o
        e         e    a    l                           o     r    r
n       c    t    n    r    a                           r     m    m
a       t    y    g    n    b                           m     a    a
m       e    p    t    u    e                           a     t    t
e       d    e    h    m    l                           t     l    d

ASL5    1    1    5    .    SERVICES LESS MEDICAL CARE        0    0
A0L5    1    1    5    .    ALL ITEMS LESS MEDICAL CARE       0    0
A5      1    1    5    .    MEDICAL CARE                      0    0
A51     1    1    5    .    MEDICAL CARE COMMODITIES          0    0
A512    1    1    5    .    MEDICAL CARE SERVICES             0    0
```

The following statements make use of this information to extract the data for A512 and descriptive information on cross sections containing A512:

```
proc format;
   value $areafmt ’0100’ = ’Northeast Region’
                  ’0200’ = ’North Central Region’
                  ’0300’ = ’Southern Region’
                  ’0400’ = ’Western Region’;
run;

filename datafile ’host-specific-file-name’ <host-options>;
proc datasource filetype=blscpi interval=month
                out=medical outall=medinfo;
   where survey=’CU’ and area in ( ’0100’,’0200’,’0300’,’0400’ );
   keep a512;
   range  from 1980:5;
   format area $areafmt.;
   rename a512=medcare;
run;

title1 ’Information on Medical Care Service’;
proc print data=medinfo;
run;
```

Output 14.2.2. Printout of the OUTALL= Data Set

```
                        Information on Medical Care Service

                                    b                     b                s
                                    a        b            y                e
   s    s                           s        a        l   s                l
   u    e                           p        s        e   e                e
   r    a   a                       t        e        n   l    n      k    c    t
O  v    s   r                       y        p        g   e    a      e    t    y
b  e    o   e                       p        e        t   c    m      p    e    p
s  y    n   a                       e        r        h   t    e      t    d    e

1  CU   U   Northeast Region        S   1982-84=100   5   1   MEDCAR  1    1    1
2  CU   U   North Central Region    S   1982-84=100   5   1   MEDCAR  1    1    1
3  CU   U   Southern Region         S   1982-84=100   5   1   MEDCAR  1    1    1
4  CU   U   Western Region          S   1982-84=100   5   1   MEDCAR  1    1    1

                                                                     e
                                          f   f         s            n
   v    b                           f     o   o         t            d
   a    l               l           o     r   r         _            _    n
   r    k               a           r     m   m         d            d    t      n
O  n    n               b           m     a   a         a            a    i      o
b  u    u               e           a     t   t         t            t    m      b
s  m    m               l           t     l   d         e            e    e      s

1  7   3479   MEDICAL CARE SERVICES       0   0   DEC1977   JUL1990   152   152
2  7   3578   MEDICAL CARE SERVICES       0   0   DEC1977   JUL1990   152   152
3  7   3677   MEDICAL CARE SERVICES       0   0   DEC1977   JUL1990   152   152
4  7   3776   MEDICAL CARE SERVICES       0   0   DEC1977   JUL1990   152   152

    n            s            a
    i            u            r
    n            r            e                 s
    r            t            a                 _          u
    a            i            n                 c          n    n
O   n            t            a                 o          i    d
b   g            l            m                 d          t    e
s   e            e            e                 e          s    c

1  123    ALL URBAN CONSUM    NORTHEAST        CUUR0100SA512         1
2  123    ALL URBAN CONSUM    NORTH CENTRAL    CUUR0200SA512         1
3  123    ALL URBAN CONSUM    SOUTH            CUUR0300SA512         1
4  123    ALL URBAN CONSUM    WEST             CUUR0400SA512         1
```

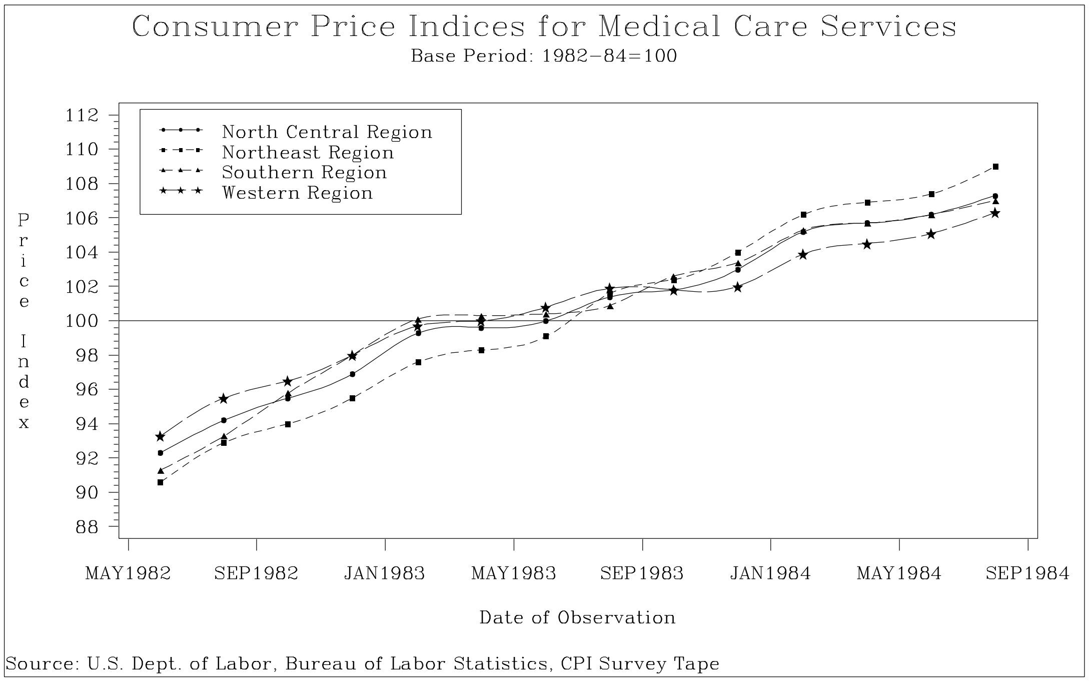
Note that only the cross sections with BASEPER='1982-84=100' are listed in the OUTALL= data set (see Output 14.2.2). This is because only those cross sections contain data for MEDCARE.

The OUTALL= data set indicates that data values are stored with one decimal place (see the NDEC variable). Therefore, they need to be rescaled, as follows:

```
data medical;
   set medical;
   medcare = medcare * 0.1;
run;
```

The variation of MEDCARE against DATE with respect to different geographic regions can be demonstrated graphically, as follows:

Output 14.2.3. Plot of Time Series in the OUT= Data Set for FILETYPE=BLSCPI



This example illustrates the following features:

- Descriptive information needed to write KEEP and WHERE statements can be obtained with an initial run of the DATASOURCE procedure.
- The OUTCONT= and OUTALL= data sets may contain information on how data values are stored, such as the precision, the units, and so on.
- The OUTCONT= and OUTALL= data sets report the new series names assigned by the RENAME statement, not the old names (see the NAME variable in Output 14.2.2).
- You can use PROC FORMAT to define formats for series or BY variables to enhance your output. Note that PROC DATASOURCE associated a permanent format, $AREAFMT., with the BY variable AREA. As a result, the formatted values are displayed in the printout of the OUTALL=MEDINFO data set (see Output 14.2.2) and in the legend created by PROC GPLOT.
- The base period for all the geographical areas is the same (BASEPER='1982-84=100') as indicated by the intersections of plots with the horizontal reference line drawn at 100. This makes comparisons meaningful.

Example 14.3. BLS State and Area, Employment, Hours and Earnings Surveys

This example illustrates how to extract specific series from a State and Area, Employment, Hours and Earnings Survey. The series to be extracted is total employment in manufacturing industries with respect to states as of March, 1990.

The State and Area, Employment, Hours and Earnings survey designates the totals for manufacturing industries by DIVISION='3', INDUSTRY='0000', and DETAIL='1'. Also, statewide figures are denoted by AREA='0000'.

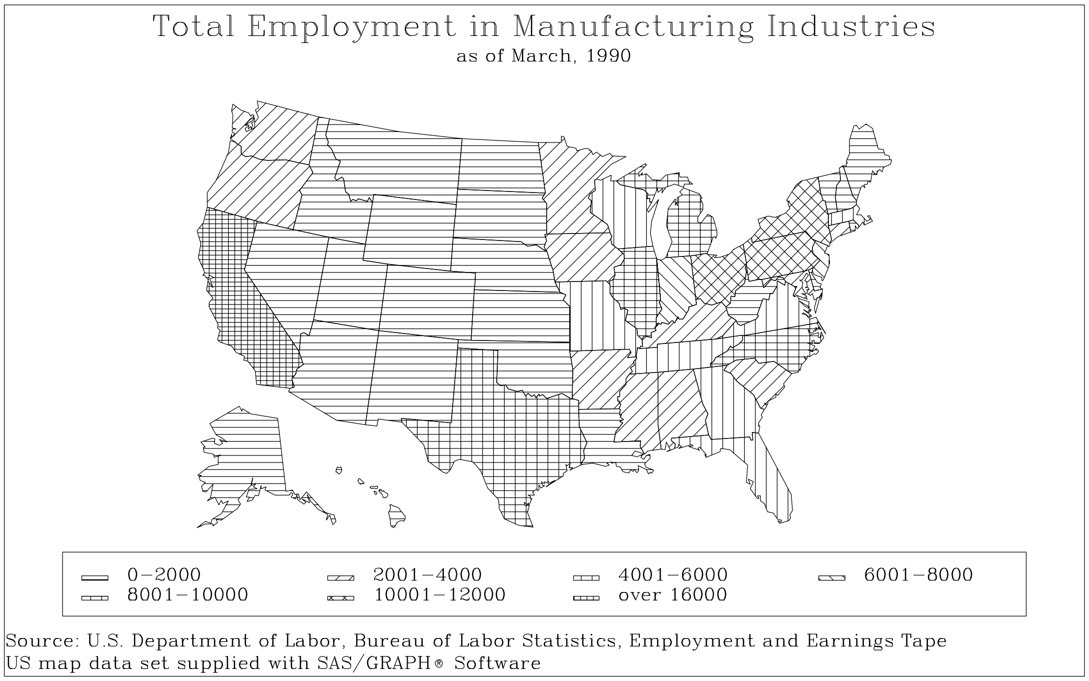
The data type code for total employment is reported to be 1. Therefore, the series name for this variable is SA1, since series names are constructed by adding an SA prefix to the data type codes given by BLS.

The following statements extract statewide figures for total employment (SA1) in manufacturing industries for March, 1990:

```
filename datafile 'host-specific-file-name' <host-options>;
proc datasource filetype=blseesa out=totemp;
   where division='3' and industry='0000' and detail='1' and
         area='0000';
   keep sa1;
   range from 1990:3 to 1990:3;
   rename sa1=totemp;
run;
```

Variations of women workers in manufacturing industries with respect to states can best be demonstrated on a map of the United States, as shown in Output 14.3.1.

Output 14.3.1. Map of the Series in the OUT= Data Set for FILETYPE=BLSEESA


Total Employment in Manufacturing Industries
as of March, 1990
0−2000
2001−4000
4001−6000
6001−8000
8001−10000
10001−12000
over 16000
Source: U.S. Department of Labor, Bureau of Labor Statistics, Employment and Earnings Tape
US map data set supplied with SAS/GRAPH® Software

Note the following for the preceding example:

- The INFILE= option is omitted, since the fileref assigned to the BLSEESA file is the default value DATAFILE.
- When the FROM and TO values in the RANGE statement are the same, only one observation for each cross section is extracted. This observation corresponds to a monthly data point since the INTERVAL= option defaults to MONTH.

Example 14.4. DRI/McGraw-Hill Tape Format CITIBASE Files

This example illustrates how to extract daily series from a sample CITIBASE file. Also, it shows how the OUTSELECT= option affects the contents of the auxiliary data sets.

The daily series contained in the sample data file CITIDEMO are listed by the following statements:

```
proc datasource filetype=citibase infile=citidemo interval=weekday
                outall=citiall outby=citikey;
run;

title1 'Summary Information on Daily Data for CITIDEMO File';
proc print data=citikey noobs;
run;

title1 'Daily Series Available in CITIDEMO File';
proc print data=citiall( drop=label );
run;
```

Output 14.4.1. Printout of the OUTBY= and OUTALL= Data Sets

```
             Summary Information on Daily Data for CITIDEMO File

   OBS      ST_DATE     END_DATE    NTIME     NOBS    NSERIES     NSELECT

    1      01JAN1988    14MAR1991     835      835       10          10
```

```
                    Daily Series Available in CITIDEMO File                 2

Obs NAME           SELECTED    TYPE    LENGTH    VARNUM    BLKNUM

  1 DSIUSNYDJCM        1         1        5         .        42
  2 DSIUSNYSECM        1         1        5         .        43
  3 DSIUSWIL           1         1        5         .        44
  4 DFXWCAN            1         1        5         .        45
  5 DFXWUK90           1         1        5         .        46
  6 DSIUKAS            1         1        5         .        47
  7 DSIJPND            1         1        5         .        48
  8 DCP05              1         1        5         .        49
  9 DCD1M              1         1        5         .        50
 10 DTBD3M             1         1        5         .        51

Obs LABEL                                                                FORMAT

  1 STOCK MKT INDEX:NY DOW JONES COMPOSITE, (WSJ)
  2 STOCK MKT INDEX:NYSE COMPOSITE, (WSJ)
  3 STOCK MKT INDEX:WILSHIRE 500, (WSJ)
  4 FOREIGN EXCH RATE WSJ:CANADA,CANADIAN $/U.S. $,NSA
  5 FOREIGN EXCH RATE WSJ:U.K.,CENTS/POUND(90 DAY FORWARD),NSA
  6 STOCK MKT INDEX:U.K. - ALL SHARES
  7 STOCK MKT INDEX:JAPAN - NIKKEI-DOW
  8 INT.RATE:5-DAY COMM.PAPER, SHORT TERM YIELD
  9 INT.RATE:1MO CERTIFICATES OF DEPOSIT, SHORT TERM YIELD (FBR H.15)
 10 INT.RATE:3MO T-BILL, DISCOUNT YIELD (FRB H.15)

Obs FORMATL   FORMATD     ST_DATE    END_DATE   NTIME   NOBS   ATTRIBUT   NDEC

  1    0         0      04JAN1988   14MAR1991    834     834       1         2
  2    0         0      04JAN1988   14MAR1991    834     834       1         2
  3    0         0      04JAN1988   14MAR1991    834     834       1         2
  4    0         0      01JAN1988   14MAR1991    835     835       1         4
  5    0         0      01JAN1988   14MAR1991    835     835       1         2
  6    0         0      01JAN1988   14MAR1991    835     835       1         2
  7    0         0      01JAN1988   14MAR1991    835     835       1         2
  8    0         0      04JAN1988   24FEB1989    300     300       2         2
  9    0         0      04JAN1988   08MAR1991    830     830       1         2
 10    0         0      04JAN1988   08MAR1991    830     830       1         2
```

Note the following from Output 14.4.1:

- The OUTALL= data set reports the time ranges of variables.
- There are ten observations in the OUTALL= data set, the same number as reported by NSERIES and NSELECT variables in the OUTBY= data set.
- The VARNUM variable contains all MISSING values, since no OUT= data set is created.

The next step is to demonstrate how the OUTSELECT= option affects the contents of the OUTBY= and OUTALL= data sets when a KEEP statement is present. First, set the OUTSELECT= option to OFF.

```
proc datasource filetype=citibase infile=citidemo interval=weekday
                outall=alloff outby=keyoff outselect=off;
   keep dsiusnysecm dc:;
run;
```

```
title1 'Summary Information on Daily Data for CITIDEMO File';
proc print data=keyoff;
run;

title1 'Daily Series Available in CITIDEMO File';
proc print data=alloff( keep=name kept selected st_date
                              end_date ntime nobs );
run;
```

Output 14.4.2. Printout of the OUTBY= and OUTALL= Data Sets with OUTSELECT=OFF

```
             Summary Information on Daily Data for CITIDEMO File

   OBS     ST_DATE    END_DATE    NTIME    NOBS    NSERIES    NSELECT

    1    01JAN1988   14MAR1991     835      834       10          3
```

```
                         Daily Series Available in CITIDEMO File

  Obs   NAME          KEPT   SELECTED     ST_DATE    END_DATE   NTIME   NOBS

    1   DSIUSNYDJCM     0        0      04JAN1988   14MAR1991    834     834
    2   DSIUSNYSECM     1        1      04JAN1988   14MAR1991    834     834
    3   DSIUSWIL        0        0      04JAN1988   14MAR1991    834     834
    4   DFXWCAN         0        0      01JAN1988   14MAR1991    835     835
    5   DFXWUK90        0        0      01JAN1988   14MAR1991    835     835
    6   DSIUKAS         0        0      01JAN1988   14MAR1991    835     835
    7   DSIJPND         0        0      01JAN1988   14MAR1991    835     835
    8   DCP05           1        1      04JAN1988   24FEB1989    300     300
    9   DCD1M           1        1      04JAN1988   08MAR1991    830     830
   10   DTBD3M          0        0      04JAN1988   08MAR1991    830     830
```

Then, set the OUTSELECT= option ON.

```
proc datasource filetype=citibase infile=citidemo interval=weekday
                outall=allon outby=keyon outselect=on;
   keep dsiusnysecm dc:;
run;

title1 'Summary Information on Daily Data for CITIDEMO File';
proc print data=keyon;
run;

title1 'Daily Series Available in CITIDEMO File';
proc print data=allon( keep=name kept selected st_date
                             end_date ntime nobs );
run;
```

Output 14.4.3. Printout of the OUTBY= and OUTALL= Data Sets with OUTSELECT=ON

```
                Summary Information on Daily Data for CITIDEMO File

     OBS       ST_DATE      END_DATE     NTIME     NOBS     NSERIES     NSELECT

      1      04JAN1988     14MAR1991      834       834        10          3
```

```
                        Daily Series Available in CITIDEMO File

   Obs    NAME           KEPT    SELECTED      ST_DATE     END_DATE    NTIME    NOBS

     1    DSIUSNYSECM      1         1       04JAN1988    14MAR1991     834      834
     2    DCP05            1         1       04JAN1988    24FEB1989     300      300
     3    DCD1M            1         1       04JAN1988    08MAR1991     830      830
```

Comparison of Output 14.4.2 and Output 14.4.3 reveals the following:

- The OUTALL= data set contains ten (NSERIES) observations when OUTSELECT=OFF, and three (NSELECT) observations when OUTSELECT=ON.
- The observations in OUTALL=ALLON are those for which SELECTED=1 in OUTALL=ALLOFF.
- The time ranges in the OUTBY= data set are computed over all the variables (selected or not) for OUTSELECT=OFF, resulting in ST_DATE='01JAN88'd and END_DATE='14MAR91'd; and over only the selected variables for OUTSELECT=ON, resulting in ST_DATE='04JAN88'd and END_DATE='14MAR91'd. This corresponds to computing time ranges over all the series reported in the OUTALL= data set.
- The variable NTIME is the number of time periods between ST_DATE and END_DATE, while NOBS is the number of observations the OUT= data set is to contain. Thus, NTIME is different depending on whether the OUTSELECT= option is set to ON or OFF, while NOBS stays the same.

Also the use of the KEEP statement in the last two examples illustrates the use of an additional variable, KEPT, in the OUTALL= data sets of Output 14.4.2 and Output 14.4.3. KEPT, which reports the outcome of the KEEP statement, is only added to the OUTALL= data set when there is KEEP statement, as shown in Output 14.4.1.

Adding the RANGE statement to the last example generates the data sets in Output 14.4.4:

```
proc datasource filetype=citibase infile=citidemo interval=weekday
                outby=keyrange out=citiday outselect=on;
   keep dsiusnysecm dc:;
   range to '12jan88'd;
run;
```

```
title1 'Summary Information

title1 'Daily Data in CITIDEMO File';
proc print data=citiday;
run;
```

Output 14.4.4. Printout of the OUT=CITIDAY Data Set for FILETYPE=CITIBASE

```
                       Daily Series Available in CITIDEMO File

OBS      ST_DATE      END_DATE     NTIME     NOBS     NINRANGE     NSERIES     NSELECT

 1     04JAN1988     14MAR1991      834       834         7           10          3
```

```
                              Daily Data in CITIDEMO File

              Obs          DATE     DSIUSNYSECM      DCP05       DCD1M

               1      04JAN1988       142.900      6.81000    6.89000
               2      05JAN1988       144.540      6.84000    6.85000
               3      06JAN1988       144.820      6.79000    6.87000
               4      07JAN1988       145.890      6.77000    6.88000
               5      08JAN1988       137.030      6.73000    6.88000
               6      11JAN1988       138.810      6.81000    6.89000
               7      12JAN1988       137.740      6.73000    6.83000
```

The OUTBY= data set in this last example contains an additional variable NINRANGE. This variable is added since there is a RANGE statement. Its value, 7, is the number of observations in the OUT= data set. In this case, NOBS gives the number of observations the OUT= data set would contain if there were not a RANGE statement.

Note that the OUT= data set does not contain data for 09JAN1988 and 10JAN1988. This is because the WEEKDAY interval skips over weekends.

Example 14.5. DRI Data Delivery Service Database

This example demonstrates the DRIDDS filetype for the daily Federal Reserve Series fxrates_dds. Use VALIDVARNAME=ANY on your SAS options statement to allow special characters such as @, $, and % to be allowed in the series name. Note the use of long variable names in the OUT= data set and long labels in the OUTCONT= data set.

The following statements extract the daily series starting from January 1,1997:

```
filename datafile 'host-specific-file-name' <host-options>;
proc format;
   value distekfm 0 = 'Unspecified'
                  2 = 'Linear'
                  4 = 'Triag'
                  6 = 'Polynomial'
                  8 = 'Even'
                 10 = 'Step'
                 12 = 'Stocklast'
                 14 = 'LinearUnadjusted'
                 16 = 'PolyUnadjusted'
                 18 = 'StockWithNAS'
                 99 = 'None'
                255 = 'None';

   value convtkfm 0 = 'Unspecified'
                  1 = 'Average'
                  3 = 'AverageX'
                  5 = 'Sum'
                  7 = 'SumAnn'
                  9 = 'StockEnd'
                 11 = 'StockBegin'
                 13 = 'AvgNP'
                 15 = 'MaxNP'
                 17 = 'MinNP'
                 19 = 'StockEndNP'
                 21 = 'StockBeginNP'
                 23 = 'Max'
                 25 = 'Min'
                 27 = 'AvgXNP'
                 29 = 'SumNP'
                 31 = 'SumAnnNP'
                 99 = 'None'
                255 = 'None';

 /*---------------------------------------------------------------*
  *                 process daily series                          *
  *---------------------------------------------------------------*/
title3 'Reading DAILY Federal Reserve Series with fxrates_.dds';
proc datasource filetype=dridds
                infile=datafile
                interval=day
                out=fixr
```

```
                     outcont=fixrcnt
                     outall=fixrall;

      range from '01jan97'd to '31dec99'd;
      format disttek distekfm.;
      format convtek convtkfm.;
   run;
```

Example 14.6. PC Diskette Format CITIBASE Database

This example uses a diskette format CITIBASE database (FILETYPE=CITIDISK) to extract annual population estimates for females and males with respect to various age groups since 1980.

Population estimate series for females with five-year age intervals are given by PANF1 through PANF16, where PANF1 is for females under 5 years of age, PANF2 is for females between 5 and 9 years of age, and so on. Similarly, PANM1 through PANM16 gives population estimates for males with five-year age intervals.

The following statements extract the required population estimates series:

```
   filename keyfile 'host-specific-key-file-name' <host-options>;
   filename indfile 'host-specific-ind-file-name' <host-options>;
   filename dbfile  'host-specific-db-file-name'  <host-options>;
   proc datasource filetype=citidisk  infile=( keyfile indfile dbfile )
                   out=popest outall=popinfo;
      keep panf1-panf16 panm1-panm16;
      range from 1980;
   run;
```

This example demonstrates the following:

- The INFILE= options lists the filerefs of the key, index, and database files, in that order.
- The INTERVAL= option is omitted since the default interval for CITIDISK type files is YEAR.

Example 14.7. Quarterly COMPUSTAT Data Files

This example shows how to extract data from a 48-quarter Compustat Database File. For COMPUSTAT data files, the series variable names are constructed by concatenating the name of the data array DATA and the column number containing the required information. For example, for quarterly files the common stock data is in column 56. Therefore, the variable name for this series is DATA56. Similarly, the series variable names for quarterly footnotes are constructed by adding the column number to the array name, QFTNT. For example, the variable name for common stock footnotes is QFTNT14 since the 14th column of the QFTNT array contains this information.

The following example extracts common stock series (DATA56) and its footnote (QFTNT14) for Computer Programming Service Companies (DNUM=7371) and Prepackaged Software Companies (DNUM=7370) whose stocks are traded over-the-counter and not in the S&P 500 Index (ZLIST=06) and whose data reside in the over-the-counter file (FILE=06).

```
filename compstat 'host-specific-Compustat-file-name' <host-options>;
proc datasource filetype=cs48qibm infile=compstat
                out=stocks outby=company;
   keep data56 qftnt14;
   rename data56=comstock  qftnt14=ftcomstk;
   label  data56='Common Stock'
          qftnt14='Footnote for Common Stock';
   where dnum in (7370,7371) and zlist=06 and file=06;
run;

/*- add company name to the out= data set    */
data stocks;
   merge stocks company( keep=dnum cnum cic coname );
   by dnum cnum cic;
run;

title1 'Common Stocks for Software Companies for 1990';
proc print data=stocks noobs;
   where date between '01jan90'd and '31dec90'd;
run;
```

The Output 14.7.1 contains a partial listing of the STOCKS data set.

Output 14.7.1. Partial Listing of the OUT=STOCKS Data Set

```
                    Common Stocks for Software Companies for 1990

DNUM   CNUM   CIC  FILE     EIN      STK  SMBL  ZLIST  XREL  FINC  SINC  state

7370  027352  103    6   54-0856778   0   AMSY    6      0     0    10     51
7370  027352  103    6   54-0856778   0   AMSY    6      0     0    10     51
7370  027352  103    6   54-0856778   0   AMSY    6      0     0    10     51
7370  027352  103    6   54-0856778   0   AMSY    6      0     0    10     51
7370  553412  107    6   73-1064024   0   MPSG    6      0     0    10     40
7370  553412  107    6   73-1064024   0   MPSG    6      0     0    10     40
7370  553412  107    6   73-1064024   0   MPSG    6      0     0    10     40
7370  553412  107    6   73-1064024   0   MPSG    6      0     0    10     40
7371  032681  108    6   41-0905408   0   ANLY    6      0     0    27     27
7371  032681  108    6   41-0905408   0   ANLY    6      0     0    27     27
7371  032681  108    6   41-0905408   0   ANLY    6      0     0    27     27
7371  032681  108    6   41-0905408   0   ANLY    6      0     0    27     27
7371  458816  105    6   04-2448936   0   IMET    6      0     0    25     25
7371  458816  105    6   04-2448936   0   IMET    6      0     0    25     25
7371  458816  105    6   04-2448936   0   IMET    6      0     0    25     25
7371  458816  105    6   04-2448936   0   IMET    6      0     0    25     25
7371  834021  107    6   04-2453033   0   SOFT    6      0     0    25     25
7371  834021  107    6   04-2453033   0   SOFT    6      0     0    25     25
7371  834021  107    6   04-2453033   0   SOFT    6      0     0    25     25
7371  834021  107    6   04-2453033   0   SOFT    6      0     0    25     25
7371  872885  108    6   13-2635899   0   TSRI    6      0     0    10     36
7371  872885  108    6   13-2635899   0   TSRI    6      0     0    10     36
7371  872885  108    6   13-2635899   0   TSRI    6      0     0    10     36
7371  872885  108    6   13-2635899   0   TSRI    6      0     0    10     36
7371  878351  105    6   41-0918564   0   TECN    6      0     0    27     27
7371  878351  105    6   41-0918564   0   TECN    6      0     0    27     27
7371  878351  105    6   41-0918564   0   TECN    6      0     0    27     27
7371  878351  105    6   41-0918564   0   TECN    6      0     0    27     27

county      date    comstock    ftcomstk    CONAME

   13     1990:1     0.11500                AMERICAN MANAGEMENT SYSTEMS
   13     1990:2     0.11600                AMERICAN MANAGEMENT SYSTEMS
   13     1990:3     0.12200                AMERICAN MANAGEMENT SYSTEMS
   13     1990:4     0.11700                AMERICAN MANAGEMENT SYSTEMS
  143     1990:1     0.42400                MPSI SYSTEMS INC
  143     1990:2     0.42400                MPSI SYSTEMS INC
  143     1990:3     0.42400                MPSI SYSTEMS INC
  143     1990:4     0.42300                MPSI SYSTEMS INC
   53     1990:1      .                     ANALYSTS INTERNATIONAL CORP
   53     1990:2      .                     ANALYSTS INTERNATIONAL CORP
   53     1990:3      .                     ANALYSTS INTERNATIONAL CORP
   53     1990:4     0.46000                ANALYSTS INTERNATIONAL CORP
   17     1990:1     0.03600                INTERMETRICS INC
   17     1990:2     0.03600                INTERMETRICS INC
   17     1990:3     0.03600                INTERMETRICS INC
   17     1990:4      .                     INTERMETRICS INC
   17     1990:1     0.38700                SOFTECH INC
   17     1990:2     0.38700                SOFTECH INC
   17     1990:3      .                     SOFTECH INC
   17     1990:4      .                     SOFTECH INC
  103     1990:1     0.02500                TSR INC
  103     1990:2     0.02500                TSR INC
  103     1990:3      .                     TSR INC
  103     1990:4      .                     TSR INC
   53     1990:1     0.21500                TECHNALYSIS CORP
   53     1990:2     0.21600                TECHNALYSIS CORP
   53     1990:3     0.21600                TECHNALYSIS CORP
   53     1990:4     0.21600                TECHNALYSIS CORP
```

Note that quarterly Compustat data are also available in Universal Character format. If you have this type of file instead of IBM 360/370 General format, use the FILETYPE=CS48QUC option instead.

Example 14.8. Annual COMPUSTAT Data Files

This example shows how to extract a subset of cross sections when the required cross sections are listed in an external file. In the case of a COMPUSTAT file, the required cross sections are a list of companies. For example, you may want to extract annual data for a list of companies whose industry classification codes (DNUM), CUSIP issuer codes (CNUM), and CUSIP issue number and check-digits (CIC) are given in an external file, COMPLIST, as follows:

```
2640   346377   104
3714   017634   106
5812   171583   107
6025   446150   104
8051   087851   101
```

When the required companies are listed in an external file, you can either use the SAS macro processor to construct your WHERE statement expression or restructure your data file and include it after the WHERE key word.

The following steps use the first approach to construct the WHERE statement expression in the macro variable WHEXPR:

```
filename compfile ’host-specific-file-name’ <host-options>;
%macro whstmt( fileref );
   %global whexpr;
   data _null_;
      infile &fileref end=last;
      length cnum $ 6;
      input  dnum cnum cic;
      call symput( ’dnum’||left(_n_), left(dnum) );
      call symput( ’cnum’||left(_n_), cnum );
      call symput( ’cic’ ||left(_n_), left(cic) );
      if last then call symput( ’n’, left(_n_) );
   run;
   %do i = 1 %to &n;
      %let whexpr = &whexpr
       (DNUM=&&dnum&i and CNUM="&&cnum&i" and CIC=&&cic&i);
      %if &i ^= &n %then %let whexpr = &whexpr or;
      %end;
   %mend whstmt;
%whstmt( compfile );
filename compustat ’host-specific-Compustat-file-name’ <host-options>;
proc datasource filetype=csaibm infile=compstat
                outby=company  out=dataset;
   where &whexpr;
run;
```

The same result can also be obtained by creating an external file, WHEXPR, from the COMPFILE and including it after the WHERE key word, as shown in the following statements:

```
filename whexpr ’host-specific-WHEXPR-file-name’ <host-options>;
data _null_;
   infile compfile end=last; file   whexpr;
   length cnum $ 6;
   input  dnum cnum cic;
   put "( "  dnum=  "and CNUM=’"  cnum $6.  "’ and "  cic= ")"  @;
   if not last then put ’ or’; else put ’;’ ;
run;

filename compstat ’host-specific-Compustat-file-name’ <host-options>;
proc datasource filetype=csaibm infile=compustat
                outby=company  out=dataset;
   where %inc ’host-specific-WHEXPR-file-name’;
run;

title1 ’Information on Selected Companies’;
proc print data=company;
run;
```

The Output 14.8.1 shows the OUTBY= data set created by the preceding statements. As you can see, the companies listed in the COMPLIST file are reported in this data set.

Output 14.8.1. Printout of the OUTBY= Data Set Listing Selected Companies

```
                       Information on Selected Companies




                                                                       C
                                Z                               S      O
     D      C              F    L     S          X              T      U      F
O    N      N        C     I    I     M          R    S         A      N      I       E
b    U      U        I     L    S     B          E    T         T      T      N       I
s    M      M        C     E    T     L          L    K         E      Y      C       N

1  2640   346377   104     3    4    FOR         0    0        34     31      0   34-1046753
2  3714   017634   106     1    4    ALN         0    0        36    103      0   38-0290950
3  5812   171583   107    11    1    CHU      5812    0        48     29      0   74-1507270
4  6025   446150   104     3    6    HBAN        0    0        39     49      0   31-0724920
5  8051   087851   101    11    1    BEV      8050    0         6     37      0   95-4100309

   b             e
   y      s      n             n      n
   s      t      d             s      s
   e      _      _     n       e      e        I
   l      d      d     t    n  r      l        N
O  e      a      a     i    o  i      e    R   A
b  c      t      t     m    b  e      c    E   M
s  t      e      e     e    s  s      t    C   E

1  1   1968   1987    20   20   423   366    1   CONVRT,PAPRBRD PD,EX CONTAIN
2  1   1968   1987    20   20   423   366    1   MOTOR VEHICLE PART,ACCESSORY
3  1   1968   1987    20   20   423   366    1   EATING PLACES
4  1   1968   1987    20   20   423   366    1   NATL BANKS-FED RESERVE SYS
5  1   1968   1987    20   20   423   366    1   SKILLED NURSING CARE FAC



   C
   O                                 D       C                    F
   N                                 N       N        C     R     I
O  A                           D     U       U        I     E     L
b  M                           U     M       M        C     C     E
s  E                           P     2       2        2     2     2

1  FORMICA CORP                0    2640    346377    104    2     3
2  ALLEN GROUP                 0    3714    017634    106    2     1
3  CHURCH'S FRIED CHICKEN INC  0    5812    171583    107    2    11
4  HUNTINGTON BANCSHARES       0    6025    446150    104    2     3
5  BEVERLY ENTERPRISES         0    8051    087851    101    2    11
```

Note that annual COMPUSTAT data are available in either IBM 360/370 General format or the Universal Character format. The first example expects an IBM 360/370 General format file since the FILETYPE= is set to CSAIBM, while the second example uses a Universal Character format file (FILETYPE=CSAUC).

Example 14.9. CRSP Daily NYSE/AMEX Combined Stocks

This example reads all the data on a three-volume daily NYSE/AMEX combined character data set. Assume that the following filerefs are assigned to the calendar/indices file and security files comprising this database:

Fileref	VOLSER	File Type
calfile	DXAA1	calendar/indices file on volume 1
secfile1	DXAA1	security file on volume 1
secfile2	DXAA2	security file on volume 2
secfile3	DXAA3	security file on volume 3

The data set CALDATA is created by the following statements to contain the calendar/indices file:

```
proc datasource filetype=crspdci infile=calfile out=caldata;
run;
```

Here the FILETYPE=CRSPDCI indicates that you are reading a character format (indicated by a C in the 6th position) daily (indicated by a D in the 5th position) calendar/indices file (indicated by an I in the 7th position).

The annual data in security files can be obtained by the following statements:

```
proc datasource filetype=crspdca
                infile=( secfile1 secfile2 secfile3 )
                out=annual;
run;
```

Similarly, the data sets to contain the daily security data (the OUT= data set) and the event data (the OUTEVENT= data set) are obtained by the following statements:

```
proc datasource filetype=crspdcs
                infile=( calfile secfile1 secfile2 secfile3 )
                out=periodic index outevent=events;
run;
```

Note that the FILETYPE= has an S at the 7th position, since you are reading the security files. Also, the INFILE= option first expects the fileref of the calendar/indices file since the dating variable (CALDT) is contained in that file. Following the fileref of calendar/indices file, you give the list of security files in the order you want to read them.

The Output 14.9.1 is generated by the following statements:

```
title1 'First 5 Observations in the Calendar/Indices File';
proc print data=caldata( obs=5 );
run;

title1 'Last 5 Observations in the Calendar/Indices File';
proc print data=caldata( firstobs=6659 ) noobs;
run;

title1 "Periodic Series for CUSIP='09523220'";
title2 "DATE >= '22dec88'd";
proc print data=periodic;
   where cusip='09523220' and date >= '22dec88'd;
run;

title1 "Events for CUSIP='09523220'";
proc print data=events;
   where cusip='09523220';
run;
```

Output 14.9.1. Partial Listing of the Output Data Sets

```
                First 5 Observations in the Calendar/Indices File

Obs        date      VWRETD      VWRETX      EWRETD      EWRETX      TOTVAL

  1   02JUL1962    -99.0000    -99.0000    -99.0000    -99.0000   319043897
  2   03JUL1962      0.0113      0.0112      0.0131      0.0130   322929231
  3   05JUL1962      0.0060      0.0059      0.0069      0.0068   324750979
  4   06JUL1962     -0.0107     -0.0107     -0.0064     -0.0064   321302641
  5   09JUL1962      0.0067      0.0067      0.0018      0.0018   323221296

Obs   TOTCNT       USDVAL    USDCNT    SPINDX      SPRTRN

  1    2036             0         0     55.86    -99.0000
  2    2040     319043897      2036     56.49      0.0113
  3    2031     322838977      2031     56.81      0.0057
  4    2031     324699079      2022     56.17     -0.0113
  5    2029     320935790      2019     56.55      0.0068
```

```
                 Last 5 Observations in the Calendar/Indices File

     date       VWRETD       VWRETX       EWRETD       EWRETX       TOTVAL

  23DEC1988    0.0042154    0.0028936     0.005104     0.003588    2367541510
  27DEC1988    -.0029128    -.0029624    -0.001453    -0.001585    2360680550
  28DEC1988    0.0015624    0.0015249     0.001575     0.001484    2364369540
  29DEC1988    0.0067816    0.0066433     0.005578     0.005469    2379932980
  30DEC1988    -.0027338    -.0029144     0.010736     0.010572    2362374030

  TOTCNT      USDVAL      USDCNT    SPINDX      SPRTRN

   2563     2360655540     2561     277.87    0.0036118
   2565     2367496320     2562     276.83    -.0037429
   2568     2360668370     2564     277.08    0.0009031
   2565     2364169480     2563     279.40    0.0083724
   2567     2379932980     2565     277.72    -.0060126
```

```
                     Periodic Series for CUSIP=’09523220’
                              DATE >= ’22dec88’d

             P   C I     H
     C       E   O S H   S              B      A                          S B
     U       R   M S E   I          D   I      S                          X X
O    S       M   P U X   C          A   D      K      P     V       R     R R
b    I       N   N N C   C          T   L      H      R     O       E     E E
s    P       O   O O D   D          E   O      I      C     L       T     T T

3 09523220 75285 0 0 1 7361 22DEC1988 15.00 15.375 15.375 54300  0.016529 . .
4 09523220 75285 0 0 1 7361 23DEC1988 15.50 15.750 15.625 17700  0.016260 . .
5 09523220 75285 0 0 1 7361 27DEC1988 15.50 15.750 15.625 10600  0.000000 . .
6 09523220 75285 0 0 1 7361 28DEC1988 15.50 15.500 15.500 10600 -0.008000 . .
7 09523220 75285 0 0 1 7361 29DEC1988 15.25 15.500 15.375  7000 -0.008065 . .
8 09523220 75285 0 0 1 7361 30DEC1988 15.00 15.250 15.000 13700 -0.024390 . .
```

```
                          Events for CUSIP=’09523220’

                P     C   I         H                            N        T
      C         E     O   S   H     S      E                     C        I
      U         R     M   S   E     I      V          D          U        C
O     S         M     P   U   X     C      E          A          S        K
b     I         N     N   N   C     C      N          T          I        E
s     P         O     O   O   D     D      T          E          P        R

1  09523220   75285   0   0   1   7361   NAMES    03MAY1988   09523220   BAW
2  09523220   75285   0   0   1   7361   DIST     18JUL1988
3  09523220   75285   0   0   1   7361   SHARES   03MAY1988
4  09523220   75285   0   0   1   7361   SHARES   30SEP1988
5  09523220   75285   0   0   1   7361   SHARES   30DEC1988
6  09523220   75285   0   0   1   7361   DELIST   30DEC1988

         C         S     E          D      D        F        D        R
         O         H  S  X    S     I      I     F  A        C        C
         M         R  H  C    I     S      V     A  C        L        R
O        N         C  R  H    C     T      A     C  S        R        D
b        A         L  C  C    C     C      M     P  H        D        D
s        M         S  D  D    D     D      T     R  R        T        T

1  BLUE ARROW PLC     3  1  7361     .   .       .  .        .        .
2                     .  .     .  1212  0.13376  0  0  13JUL88  22JUL88
3                     .  .     .     .   .       .  .        .        .
4                     .  .     .     .   .       .  .        .        .
5                     .  .     .     .   .       .  .        .        .
6                     .  .     .     .   .       .  .        .        .

                S     S    D    N         N                       T   N       N
         P      H     H    L    W         E   D   D   D   D   D   R   M   M   S
         A      R     R    S    P         X   L   L   L   L   L   T   S   M   D
O        Y      O     F    T    E         T   B   A   P   V   R   S   I   C   I
b        D      U     L    C    R         D   I   S   R   O   E   C   N   N   N
s        T      T     G    D    M         T   D   K   C   L   T   D   D   T   X

1        .        .   .     .   .         .   .   .   .   .   .   .   .   .   .
2  26AUG88        .   .     .   .         .   .   .   .   .   .   .   .   .   .
3        .    72757   0     .   .         .   .   .   .   .   .   .   .   .   .
4        .   706842   0     .   .         .   .   .   .   .   .   .   .   .   .
5        .   706842   0     .   .         .   .   .   .   .   .   .   .   .   .
6        .        .   .   100   0         .   .   .   0   .   A   .   .   .   .
```

This example illustrates the following points:

- When data span more than one physical volume, the filerefs of the security files residing on each volume must be given following the fileref of the calendar/indices file. The DATASOURCE procedure reads each of these files in the order they are specified. Therefore, you can request that all three volumes be mounted to the same tape drive, if you choose to do so.
- The INDEX option in the second PROC DATASOURCE run creates an index file for the OUT=PERIODIC data set. This index file provides random access to the OUT= data set and may increase the efficiency of the subsequent PROC and DATA steps that use BY and WHERE statements. The index variables are CUSIP, CRSP permanent number (PERMNO), NASDAQ company number (COMPNO), NASDAQ issue number (ISSUNO), header exchange code (HEXCD) and header SIC code (HSICCD). Each one of these variables forms a different key, that is, a single index. If you want to form keys from a combination of variables (composite indexes) or use some other variables as indexes, you should use the INDEX= data set option for the OUT= data set.
- The OUTEVENT=EVENTS data set is sparse. In fact, for each EVENT type, a unique set of event variables are defined. For example, for EVENT='SHARES', only the variables SHROUT and SHRFLG are defined, and they have missing values for all other EVENT types. Pictorially, this structure is similar to the data set shown in Figure 14.8. Because of this sparse representation, you should create the OUTEVENT= data set only when you need a subset of securities and events.

By default, the OUT= data set contains only the periodic data. However, you may also want to include the event-oriented data in the OUT= data set. This is accomplished by listing the event variables together with periodic variables in a KEEP statement. For example, if you want to extract the historical CUSIP (NCUSIP), number of shares outstanding (SHROUT), and dividend cash amount (DIVAMT) together with all the periodic series, use the following statements:

```
proc datasource filetype=crspdcs
                infile=( calfile secfile1 secfile2 secfile3 )
                out=both outevent=events;
   where cusip='09523220';
   keep  bidlo askhi prc vol ret sxret bxret ncusip shrout divamt;
run;

proc datasource filetype=crspdcs
                infile=( calfile secfile1 )
                out=both outevent=events;
   where cusip='09523220';
   keep  bidlo askhi prc vol ret sxret bxret ncusip shrout divamt;
run;

proc datasource filetype=crspdcs
                infile=( calfile secfile1 )
                out=both2 outevent=events2;
   where cusip='09523220';
   keep  bidlo askhi prc vol ret sxret bxret ncusip shrout divamt;
   keepevent ncusip shrflg;
```

```
run;

title1 "Printout of the First 4 Observations";
title2 "CUSIP = ’09523220’";
proc print data=both noobs;
   var  cusip date vol ncusip divamt shrout;
   where cusip=’09523220’ and date <= ’08may88’d;
run;

title1 "Printout of the Observations centered Around 18jul88";
title2 "CUSIP = ’09523220’";
proc print data=both noobs;
   var  cusip date vol ncusip divamt shrout;
   where cusip=’09523220’ and
         date between ’14jul88’d and ’20jul88’d;
run;

title1 "Printout of the Observations centered Around 30sep88";
title2 "CUSIP = ’09523220’";
proc print data=both noobs;
   var  cusip date vol ncusip divamt shrout;
   where cusip=’09523220’ and
         date between ’28sep88’d and ’04oct88’d;
run;
```

Output 14.9.2. Including Event Variables in the OUT= Data Set

```
                    Printout of the First 4 Observations
                            CUSIP = ’09523220’

       CUSIP            DATE       VOL       NCUSIP      DIVAMT    SHROUT

     09523220    03MAY1988    296100    09523220        .        72757
     09523220    04MAY1988    139200    09523220        .        72757
     09523220    05MAY1988      9000    09523220        .        72757
     09523220    06MAY1988      7900    09523220        .        72757
```

```
            Printout of the Observations centered Around 18jul88
                            CUSIP = ’09523220’

       CUSIP            DATE       VOL       NCUSIP       DIVAMT    SHROUT

     09523220    14JUL1988     62000    09523220     .            72757
     09523220    15JUL1988    106800    09523220     .            72757
     09523220    18JUL1988     32100    09523220    0.13376       72757
     09523220    19JUL1988      8600    09523220     .            72757
     09523220    20JUL1988     10700    09523220     .            72757
```

```
            Printout of the Observations centered Around 30sep88
                            CUSIP = ’09523220’

       CUSIP            DATE       VOL       NCUSIP      DIVAMT    SHROUT

     09523220    28SEP1988     33000    09523220        .        72757
     09523220    29SEP1988     55200    09523220        .        72757
     09523220    30SEP1988     40700    09523220        .       706842
     09523220    03OCT1988     13400    09523220        .       706842
     09523220    04OCT1988    110600    09523220        .       706842
```

Events referring to distributions and delistings have entries only in observations whose dates match the event dates. For example, DIVAMT has a value for only 18JUL88, as shown in the second printout in Output 14.9.2. The NAME and SHARES events refer to a date of change, therefore their values are expanded such that there is a value for each observation. For example, the date of NAMES record is 03MAY88, therefore NCUSIP has the same value from that date on. The SHROUT on the other hand changes its value twice, once on 03MAY88, the other time on 30SEP88. The third listing shows how the value of SHROUT remains constant at 72757 from 03MAY88 to 30SEP88, at which date it changes to 706842.

The events occurring on days other than the trading dates are not output to the OUT= data set.

The KEEP statement in the preceding example has no effect on the event variables output to the OUTEVENT= data set. If you want to extract only a subset of event variables, you need to use the KEEPEVENT statement. For example, the following code outputs only NCUSIP and SHROUT to the OUTEVENT= data set for CUSIP='09523220':

```
proc datasource filetype=crspdxc
                infile=( calfile secfile1 secfile2 secfile3 )
                outevent=subevts;
   where cusip='09523220';
   keepevent  ncusip shrout;
run;

proc datasource filetype=crspdxc
                infile=( calfile secfile1)
                outevent=subevts;
   where cusip='09523220';
   keepevent  ncusip shrout;
run;

title1 "NCUSIP and SHROUT for CUSIP='09523220'";
proc print data=subevts noobs;
run;
```

Output 14.9.3. Listing of the OUTEVENT= Data Set with a KEEPEVENT Statement

```
                     NCUSIP and SHROUT for CUSIP='09523220'

  CUSIP   PERMNO COMPNO ISSUNO HEXCD HSICCD EVENT       DATE  NCUSIP   SHROUT

09523220  75285    0      0      1    7361  NAMES  03MAY1988 09523220      .
09523220  75285    0      0      1    7361  SHARES 03MAY1988            72757
09523220  75285    0      0      1    7361  SHARES 30SEP1988           706842
09523220  75285    0      0      1    7361  SHARES 30DEC1988           706842
```

The OUTEVENT= data set in Output 14.9.3 is missing observations for which the EVENT variable is DIST or DELIST, since these event groups do not contain any selected events.

Example 14.10. CRSP 1995 CDROM Data Files

The normal character filetypes used for tape files may also be used for the CDROM character data. They are CRSPDCS, CRSPDCI, CRSPDCA for daily data and CRSPMCS, CRSPMCI, CRSPMCA for monthly data. It is necessary to use the LRECL=(130 401) option on the DATASOURCE statement when processing CDROM character data as shown in the first DATASOURCE run.

The CRSP 1995 UNIX (SUN)Binary data is readable by PROC DATASOURCE using the filetypes CRSPDUS, CRSPDUI, CRSPDUA for daily files and filetypes CRSPMUS, CRSPMUI, CRSPMUA for monthly files. Both IEEE Big Endian and IEEE Little Endian machines may use the same CDROM UNIX Binary filetypes. PROC DATASOURCE can not read the PC Binary Data from CDROM, but the UNIX (SUN) Binary may be used from the same CDROM instead, even on PC's. The second DATASOURCE run shows how to access the 1995 UNIX Binary data.

```
filename csec 'machar.dat' recfm=f lrecl=401;
filename ccal 'msix.dat' recfm=f lrecl=130;

/*-------------------------------------------------------------*
 *       create output data sets without any subsetting        *
 *       character data from MA CDROM                          *
 *-------------------------------------------------------------*/
/*- create calendar/indices output data sets using DATASOURCE  -*/
/*- statements                                                 -*/
proc datasource filetype=crspmcs
               infile=( ccal csec )
               lrecl=( 130 401 )
               interval=month
               outselect=off
               outcont=maccont outkey=mackey
               out=mac outevent=macevent;
   keep _all_;
   keepevent _all_;
run;

title3 'MA/CDROM Security File Outputs';
title4 'OUTKEY= Data Set';
proc print data=mackey; run;

title4 'OUTCONT= Data Set';
proc print data=maccont; run;

title4 'Listing of OUT= Data Set';
proc print data=mac; run;

title4 'Listing of OUTEVENT= Data Set';
proc print data=macevent; run ;

filename macal 'maucal95.data' lrecl=48;
filename masec 'mausub95.data' recfm=v lrecl=32760;

/*------------------------------------------------------------*
 *       create output data sets without any subsetting       *
 *       UNIX (SUN) binary data from MA CDROM                 *
 *------------------------------------------------------------*/
```

```
/*- create calendar/indices output data sets using DATASOURCE -*/
/*- statements                                                 -*/
proc datasource filetype=crspmus
               infile=( macal masec )
               interval=month
               outselect=off
               outcont=macont outkey=makey
               out=ma outevent=maevent;
   keep _all_;
   keepevent _all_;
run;

title3 'MA/CDROM Security File Outputs';
title4 'OUTKEY= Data Set';
proc print data=makey; run;

title4 'OUTCONT= Data Set';
proc print data=macont; run;

title4 'Listing of OUT= Data Set';
proc print data=ma; run;

title4 'Listing of OUTEVENT= Data Set';
proc print data=maevent; run ;
```

Example 14.11. CRSP ACCESS97 CDROM Data Files

This example demonstrates how to work with the CRSP ACCESS97 CDROM data files by first running the CRSP supplied *stk_dump_bin* utility, to create a UNIX (SUN) binary file. The UNIX binary file can then be processed by PROC DATASOURCE using the CRSPMUS filetype for monthly data or the CRSPDUS filetype for DAILY data.

The DATASOURCE procedure expects the data file to use IEEE big Endian byte ordering. The exact command that you need to use to convert the data depends on whether you extracted the big Endian or little Endian data off of the CD, and whether you are running on a host whose native binary representation is big or little Endian. Consult your *1997 CRSP ACCESS97 Stock File User's Guide*, *Appendix C* for details on the reverse/keep option for the byte-ordering flag. Assuming a Windows NT platform and daily data:

```
ind_dump_bin %crsp_dstk% filename1 460 1000080 1000081 1000502 reverse unix
stk_dump_bin %crsp_dstk% filename2 10 1 0 0 0 reverse unix permlist_filename
```

Once you have converted the ACCESS97 data into the unix binary format, you are ready to invoke PROC DATASOURCE:

```
filename calfile 'filename1';
filename secfile 'filename2' lrecl=36000;

proc datasource filetype=crspdus
   infile=( calfile secfile )
   interval=day
```

```
      outselect=off
      out=da outkey=dakey outcont=dacont outevent=daevent;
      keep _all_;
      keepevent _all_;
   run;
```

The above example uses an LRECL to accommodate the size of the 1997 daily data. Subsequent years may need a larger lrecl.

Example 14.12. IMF Direction of Trade Statistics

This example illustrates how to extract data from a Direction of Trade Statistics (DOTS) data file. The DOTS data files contain only two series, EXPORTS and IMPORTS, for various sets of countries. The foreign trade figures between any two countries can be extracted by specifying their three-digit codes for COUNTRY and PARTNER BY variables. The following statements can then be used to extract quarterly EXPORTS and IMPORTS between the United States of America (COUNTRY='111') and Japan (PARTNER='158').

```
filename dotsfile 'host-specific-gfs-file-name' <host-options>;
proc datasource filetype=imfdotsp infile=dotsfile interval=qtr
                out=foreign outall=forngvar;
   where country='111' and partner='158';
run;
```

References

Bureau of Economic Analysis (1986), *The National Income and Product Accounts of the United States, 1929-82*, U.S. Dept of Commerce, Washington D.C.

Bureau of Economic Analysis (1987), *Index of Items Appearing in the National Income and Product Accounts Tables*, U.S. Dept of Commerce, Washington D.C.

Bureau of Economic Analysis (1991), *Survey of Current Business*, U.S. Dept of Commerce, Washington D.C.

Center for Research in Security Prices (2000), *CRSP Data Description Guide*, Chicago, IL.

Center for Research in Security Prices (2000), *CRSP Programmer's Guide*, Chicago, IL.

Center for Research in Security Prices (2000), *CRSP Utilities Guide*, Chicago, IL.

Center for Research in Security Prices (2000), *CRSP SFA Guide*, Chicago, IL.

Citibank (1990), *CITIBASE Directory*, New York, NY.

Citibank (1991), *CITIBASE-Weekly*, New York, NY.

Citibank (1991), *CITIBASE-Daily*, New York, NY.

DRI/McGraw-Hill (1997), *DataLink*, Lexington, MA.

DRI/McGraw-Hill Data Search and Retrieval for Windows (1996), *DRIPRO User's Guide*, Lexington, MA.

FAME Information Services (1995), *User's Guide to FAME*, Ann Arbor, Michigan

International Monetary Fund (1984), *IMF Documentation on Computer Tape Subscription*, Washington, D.C.

Organization For Economic Cooperation and Development (1992) *Annual National Accounts: Volume I. Main Aggregates Content Documentation for Magnetic Tape Subscription*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Annual National Accounts: Volume II. Detailed Tables Technical Documentation for Magnetic Tape Subscription*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Main Economic Indicators Database Note*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Main Economic Indicators Inventory*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Main Economic Indicators OECD Statistics on Magnetic Tape Document*, Paris, France.

Organization For Economic Cooperation and Development (1992) *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Quarterly National Accounts Inventory of Series Codes*, Paris, France.

Organization For Economic Cooperation and Development (1992) *Quarterly National Accounts Technical Documentation*, Paris, France.

Standard & Poor's Compustat Services Inc. (1991), *COMPUSTAT II Documentation*, Englewood, CO.

Chapter 15

The ENTROPY Procedure (Experimental)

Chapter Contents

Chapter 15
The ENTROPY Procedure (Experimental)

Overview

The ENTROPY procedure implements a parametric method of linear estimation based on Generalized Maximum Entropy. The ENTROPY procedure is suitable when there are outliers in the data and robustness is required, or when the model is ill-posed or underdetermined for the observed data, or for regressions involving small data sets.

The main features of the ENTROPY procedure are as follows:

- estimates simultaneous systems of linear regression models
- estimates Markov models
- estimates Seemingly Unrelated Regression (SUR) models
- estimates Unordered Multinomial Discrete Choice models
- solves pure inverse problems
- allows bounds and restrictions on parameters
- performs tests on parameters
- allows data and moment constrained generalized cross entropy

Experimental graphics are now available with the ENTROPY procedure. For more information, see the "ODS Graphics" section on page 774.

It is often the case that the statistical-economic model of interest is ill-posed or underdetermined for the observed data. For the general linear model this can imply that high degrees of collinearity exist among explanatory variables or that there are more parameters to estimate than observations to estimate them with. These conditions lead to high variances or nonestimability for traditional Generalized Least Squares (GLS) estimates.

Under these situations it may be in the researcher's or practitioner's best interest to consider a nontraditional technique for model fitting. The principle of maximum entropy is the foundation for an estimation methodology that is characterized by its robustness to ill-conditioned designs and its ability to fit over-parameterized models. See Mittelhammer, Judge, and Miller (2000), and Golan, Judge, and Miller (1996) for a discussion of Shannon's maximum entropy measure and the related Kullback-Leibler information.

Generalized Maximum Entropy (GME) is a means of selecting among probability distributions so as to choose the distribution that maximizes uncertainty or uniformity remaining in the distribution, subject to information already known about the distribution. Information takes the form of data or moment constraints in the estimation procedure. PROC ENTROPY creates a GME distribution for each parameter in the linear model, based upon support points supplied by the user. The mean of each distribution is used as the estimate of the parameter. Estimates tend to be biased, as they are a type of shrinkage estimate, but will typically portray smaller variances than OLS counterparts, making them more desirable from a mean squared error viewpoint (see Figure 15.1).

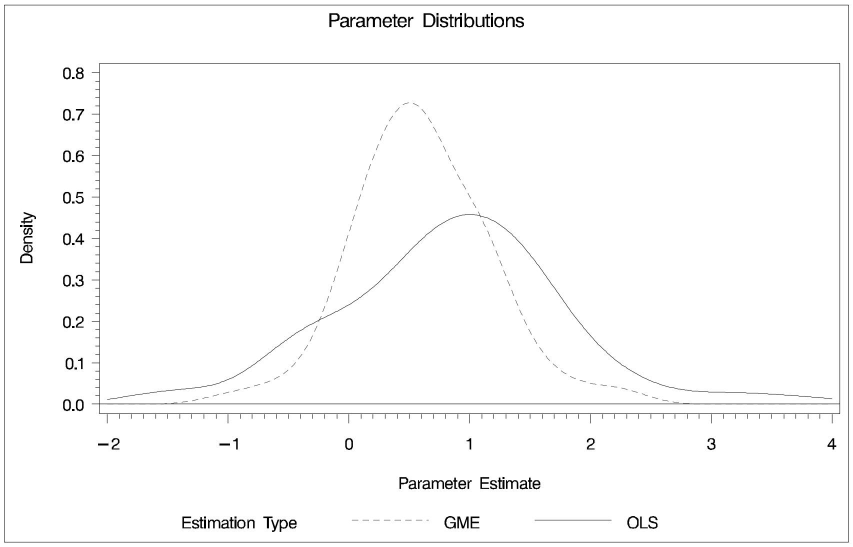


Figure 15.1. Distribution of Maximum Entropy Estimates Versus OLS

Maximum entropy techniques are most widely used in the econometric and time series fields. Some important uses of maximum entropy include

- size distribution of firms
- stationary Markov Process
- Social Accounting Matrix (SAM)
- consumer brand preference
- exchange rate regimes
- wage-dependent firm relocation
- oil market dynamics

Getting Started

This section introduces the ENTROPY procedure and shows how to use PROC ENTROPY for several kinds of regression analyses.

Simple Regression Analysis

The ENTROPY procedure is similar in syntax to the other regression procedures in the SAS System. To demonstrate the similarity, suppose the endogenous/dependent variable is `y`, and `x1` and `x2` are two exogenous/independent variables of interest. To estimate the parameters in this single equation model using PROC ENTROPY, use the following SAS statements:

```
proc entropy;
   model y = x1 x2 ;
run;
```

Test Scores Data Set

Consider the following test score data compiled by Coleman et al. (1966) :

```
Title "Test Scores compiled by Coleman et al. (1966)";
data coleman;

   input test_score 6.2 teach_sal 6.2 prcnt_prof
         8.2 socio_stat 9.2 teach_score 8.2 mom_ed 7.2;

   label test_score="Average sixth grade test scores in observed district";
   label teach_sal="Average teacher salaries per student ($1000)";
   label prcnt_prof="Percent of students’ fathers with professional employment";
   label socio_stat="Composite measure of socio-economic status in the district"
;
   label teach_score="Average verbal score for teachers";
   label mom_ed="Average level of education (years) of the students’ mothers";

datalines;
37.01   3.83   28.87     7.20   26.60   6.19
26.51   2.89   20.10   -11.71   24.40   5.17
36.51   2.86   69.05    12.32   25.70   7.04
40.70   2.92   65.40    14.28   25.70   7.10
37.10   3.06   29.59     6.31   25.40   6.15
33.90   2.07   44.82     6.16   21.60   6.41
41.80   2.52   77.37    12.70   24.90   6.86
33.40   2.45   24.67    -0.17   25.01   5.78
41.01   3.13   65.01     9.85   26.60   6.51
37.20   2.44    9.99    -0.05   28.01   5.57
23.30   2.09   12.20   -12.86   23.51   5.62
35.20   2.52   22.55     0.92   23.60   5.34
34.90   2.22   14.30     4.77   24.51   5.80
33.10   2.67   31.79    -0.96   25.80   6.19
22.70   2.71   11.60   -16.04   25.20   5.62
39.70   3.14   68.47    10.62   25.01   6.94
31.80   3.54   42.64     2.66   25.01   6.33
31.70   2.52   16.70   -10.99   24.80   6.01
```

```
   43.10   2.68   86.27    15.03   25.51   7.51
   41.01   2.37   76.73    12.77   24.51   6.96
   ;
```

This data set contains outliers and the condition number of the matrix of regressors, X, is large. Since the Entropy estimates are both robust with respect to the outliers and less sensitive to a high condition number of the X matrix, maximum entropy estimation is a good choice for this problem.

To fit a simple linear model to this data using PROC ENTROPY use the following statements:

```
   proc entropy data=coleman;
      model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
   run;
```

This requests the estimation of a linear model for TEST_SCORE with the following form

$$\begin{aligned} test\_score &= a * teach\_sal + b * prcnt\_prof + c * socio\_stat \\ &\quad + d * teach\_score + e * mom\_ed + intercept + \epsilon; \end{aligned}$$

This estimation produces the Model Summary table in Figure 15.2 showing the equation variables used in the estimation.

```
                 Test Scores compiled by Coleman et al. (1966)

                             The ENTROPY Procedure

 Variables(Supports(Weights))  teach_sal prcnt_prof socio_stat
                               teach_score mom_ed Intercept
 Equations(Supports(Weights))  test_score
```

Figure 15.2. Model Summary Table

The next four tables displayed are the Data Set Options table, the Minimization Summary, the Final Information Measures table, and the Observations Processed tables, shown in Figure 15.3.

```
                         GME-NM Estimation Summary

                              Data Set Options

                            DATA=    WORK.COLEMAN


                             Minimization Summary

                       Parameters Estimated           6
                       Covariance Estimator      GME-NM


                           Final Information Measures

                   Entropy                          9.553698
                   Parameter Information Index      0.009024
                   Error Information Index          0.000214


                            Observations Processed

                                  Read    20
                                  Used    20
```

Figure 15.3. Estimation Summary Tables

The item labeled "Entropy" is the value of the entropy estimation criterion for this estimation problem. This measure is analogous to the log-likelihood value in a maximum likelihood estimation. The "Parameter Information Index" and the "Error Information Index" are normalized entropy values that measure the proximity of the solution to the prior or target distributions.

The next table displayed is the ANOVA table, shown in Figure 15.4. This is in the same form as the ANOVA table for the MODEL procedure, since this is also a multivariate procedure.

```
                    GME-NM Summary of Residual Errors

                DF      DF
Equation     Model   Error       SSE       MSE  Root MSE  R-Square  Adj RSq

test_score       6      14     175.7    8.7866    2.9642    0.7267   0.6291
```

Figure 15.4. Summary of Residual Errors

The last table displayed is the "Parameter Estimates" table, shown in Figure 15.5. The difference between this parameter estimates table and the parameter estimates table produced by other regression procedures is that the standard error and the probabilities are labeled as approximate.

```
                         GME-NM Variable Estimates

                                       Approx                     Approx
           Variable         Estimate     Std Err     t Value      Pr > |t|

           teach_sal        0.287969     0.00551       52.27        <.0001
           prcnt_prof        0.02266     0.00323        7.01        <.0001
           socio_stat       0.199787      0.0308        6.48        <.0001
           teach_score      0.497114      0.0180       27.61        <.0001
           mom_ed           1.644449      0.0921       17.85        <.0001
           Intercept        10.50156      0.3958       26.53        <.0001
```

Figure 15.5. Parameter Estimates

The parameter estimates produced by the REG procedure for this same model are shown in Figure 15.6. Note that the parameters and standard errors from PROC REG are much different than estimates produced by PROC ENTROPY.

```
                              The REG Procedure
                                Model: MODEL1
                       Dependent Variable: test_score '

                             Parameter Estimates

                                      Parameter      Standard
  Variable       Label        DF       Estimate         Error    t Value    Pr > |t|

  Intercept      Intercept     1       19.94857      13.62755       1.46      0.1653
  teach_sal      '             1       -1.79333       1.23340      -1.45      0.1680
  prcnt_prof     '             1        0.04360       0.05326       0.82      0.4267
  socio_stat     '             1        0.55576       0.09296       5.98      <.0001
  teach_score    '             1        1.11017       0.43377       2.56      0.0227
  mom_ed         '             1       -1.81092       2.02739      -0.89      0.3868
```

Figure 15.6. REG Procedure Parameter Estimates

This data set contains two outliers, observations 3 and 18. These can be seen in a plot of the residuals shown in Figure 15.7

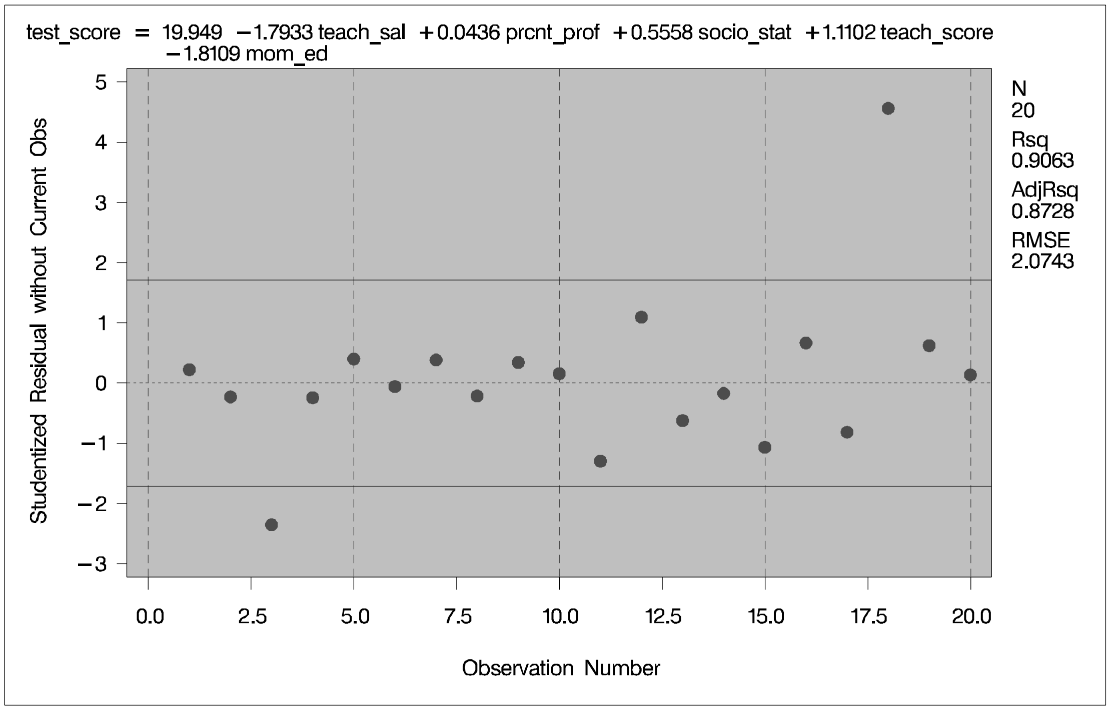


Figure 15.7. Residuals with Outliers

The presence of outliers suggests that a robust estimator such as M-estimator in the ROBUSTREG procedure should be used. The following statements use the ROBUSTREG procedure to estimate the model.

```
proc robustreg data=coleman;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;
```

The results of the estimation are shown in Figure 15.8.

```
                         The ROBUSTREG Procedure

                           Parameter Estimates

                             Standard   95% Confidence     Chi-
 Parameter    DF Estimate     Error        Limits        Square Pr > ChiSq

 Intercept     1  29.3416    6.0381  17.5072  41.1761    23.61     <.0001
 teach_sal     1  -1.6329    0.5465  -2.7040  -0.5618     8.93     0.0028
 prcnt_prof    1   0.0823    0.0236   0.0361   0.1286    12.17     0.0005
 socio_stat    1   0.6653    0.0412   0.5846   0.7461   260.95     <.0001
 teach_score   1   1.1744    0.1922   0.7977   1.5510    37.34     <.0001
 mom_ed        1  -3.9706    0.8983  -5.7312  -2.2100    19.54     <.0001
 Scale         1   0.6966
```

Figure 15.8. M-Estimation Results

Note that TEACH_SAL(VAR1) and MOM_ED(VAR5) change greatly when the robust estimation is used. Unfortunately, these two coefficients are negative, which implies that the test scores increase with decreasing teacher salaries and decreasing levels of the mother's education. Since ROBUSTREG is robust to outliers, they are not causing the counter-intuitive parameter estimates.

The condition number of the regressor matrix X also plays a important role in parameter estimation. The condition number of the matrix can be obtained by specifying the COLLIN option on the PROC ENTROPY statement.

```
proc entropy data=coleman collin;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;
```

The output produced by the COLLIN option is shown in Figure 15.9.

```
                               The ENTROPY Procedure

                              Collinearity Diagnostics

                                    ----------Proportion of Variation---------
                         Condition                prcnt_      socio_      teach_
Number     Eigenvalue       Number   teach_sal      prof        stat       score

     1       4.978128       1.0000      0.0007    0.0012      0.0026      0.0001
     2       0.937758       2.3040      0.0006    0.0028      0.2131      0.0001
     3       0.066023       8.6833      0.0202    0.3529      0.6159      0.0011
     4       0.016036      17.6191      0.7961    0.0317      0.0534      0.0059
     5       0.001364      60.4112      0.1619    0.3242      0.0053      0.7987
     6       0.000691      84.8501      0.0205    0.2874      0.1096      0.1942

                              Collinearity Diagnostics

                                   Condition   -Proportion of Variation-
          Number     Eigenvalue       Number     mom_ed    Intercept

               1       4.978128       1.0000     0.0001       0.0000
               2       0.937758       2.3040     0.0000       0.0001
               3       0.066023       8.6833     0.0000       0.0003
               4       0.016036      17.6191     0.0083       0.0099
               5       0.001364      60.4112     0.3309       0.0282
               6       0.000691      84.8501     0.6607       0.9614
```

Figure 15.9. Collinearity Diagnostics

The condition number of the X matrix is reported to be 84.85. This means that the condition number of $X'X$ is $84.85^2 = 7199.5$, which is very large.

Ridge regression can be used to offset some of the problems associated with ill-conditioned X matrices. Using the formula for the ridge value as

$$\lambda_R = \frac{kS^2}{\hat{\beta}'\hat{\beta}} \approx .9$$

where $\hat{\beta}$ and S^2 are the least-squares estimators of β and σ^2 and $k = 6$. 0.9 a ridge regression of the test score model was performed using the data set with the outliers removed. The following PROC REG code performs the ridge regression:

```
proc reg data=coleman ridge=0.9 outest=t noprint;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;
proc print data=t;run;
```

The results of the estimation are shown in Figure 15.10.

```
Obs    _MODEL_    _TYPE_     _DEPVAR_     _RIDGE_    _PCOMIT_     _RMSE_    Intercept

 1     MODEL1     PARMS     test_score       .           .       0.78236     29.7577
 2     MODEL1     RIDGE     test_score      0.9          .       3.19679      9.6698

         teach_       prcnt_       socio_      teach_                 test_
Obs        sal         prof         stat       score       mom_ed     score

 1     -1.69854     0.085118     0.66617     1.18400    -4.06675      -1
 2     -0.08892     0.041889     0.23223     0.60041     1.32168      -1
```

Figure 15.10. Ridge Regression Estimates

Note that the ridge regression estimates are much closer to the estimates produced by the ENTROPY procedure using the original data set.

Using Prior Information

You can use prior information about the parameters or the residuals to improve the efficiency of the estimates. Some authors prefer the terms *pre-sample* or *pre-data* over the term *prior* when used with Maximum Entropy to avoid confusion with Bayesian methods. The Maximum Entropy method described here does not use Bayes' rule when including prior information in the estimation.

To perform regression, the ENTROPY procedure uses generalization of Maximum Entropy called *Generalized Maximum Entropy*. In Maximum Entropy estimation the unknowns are probabilities. Generalized maximum entropy expands the set of problems that can be solved by introducing the concept of *Support points*. Generalized maximum entropy still estimates probabilities, but these are the probabilities of a support point. Support points are used to map the zero one domain of the maximum entropy to the any finite range of values.

Prior information, such as expected ranges for the parameters or the residuals, is added by specifying support points for the parameters or the residuals. Support points are points in one dimension that specify the expected domain of the parameter or the residual. The wider the domain specified, the less efficient (more variance) your parameter estimates will be. Specifying more support points in the same width interval also improves the efficiency of the parameter estimates at the cost of more computation. Golan, Judge, and Miller (1996) show that the gains in efficiency fall off for adding more than five support points. You can specify between 2 to 256 support points in the ENTROPY procedure.

If you have only a small amount of data, the estimates will be very sensitive to your selection of support points and weights. For larger data sets incorrect priors are discounted if they are not supported by the data.

Consider the data set generated by the following SAS statements:

```
data prior;
    do by = 1 to 100;
       do t = 1 to 10;
          y = 2*t + 5 * rannor(456);
          output;
       end;
    end;
run;
```

The PRIOR data set contains 100 samples of 10 observations each from the population

$$\begin{aligned} y &= 2 * t + \epsilon \\ \epsilon &\sim N(0, 5) \end{aligned}$$

You can estimate these samples using PROC ENTROPY as

```
proc entropy data = prior outest=parm1;
   model y = t ;
   by by;
run;
```

The 100 estimates are summarized using the following SAS statements:

```
proc univariate data=parm1 ;
  var t;
run;
```

The summary statistics from PROC UNIVARIATE are shown in Figure 15.11. The true value of the coefficient t is 2.0, demonstrating that maximum entropy estimates tend to be biased.

```
                     The UNIVARIATE Procedure
                           Variable:  t

                    Basic Statistical Measures

          Location                    Variability

      Mean     1.674803     Std Deviation            0.32418
      Median   1.708555     Variance                 0.10509
      Mode      .           Range                    1.80200
                            Interquartile Range      0.34135
```

Figure 15.11. No Prior Information Monte Carlo Summary

Now assume that you have prior information about the slope and the intercept for this model. You are reasonably confident that the slope is 2 and you are less confident that intercept is zero. To specify prior information on the parameters use the PRIORS statement. There are two parts to the prior information specified in the PRIORS statement. The first part is the support points for a parameter. The support points specify the domain of the parameter. For example

```
priors t -1000 1000;
```

sets the support points -1000 and 1000 for the parameter associated with variable t. This means that the coefficient lies in the interval $[-1000, 1000]$. If the estimated value of the coefficient is actually outside of this interval, the estimation will not converge. In the previous PRIORS statement, no weights were specified for the support points, so uniform weights are assumed. This implies that the coefficient has a uniform probability of being in the interval $[-1000, 1000]$.

The second part of the prior information is the weights on the support points. For example

```
   priors t 10(1) 15(5) 20(5) 25(1);
```

sets the support points 10, 15, 20, and 25 with weights 1, 5, 5, and 1 respectively for the coefficient of t. This creates the prior distribution on the coefficient shown in Figure 15.12. The weights are automatically normalized so that they sum to one.

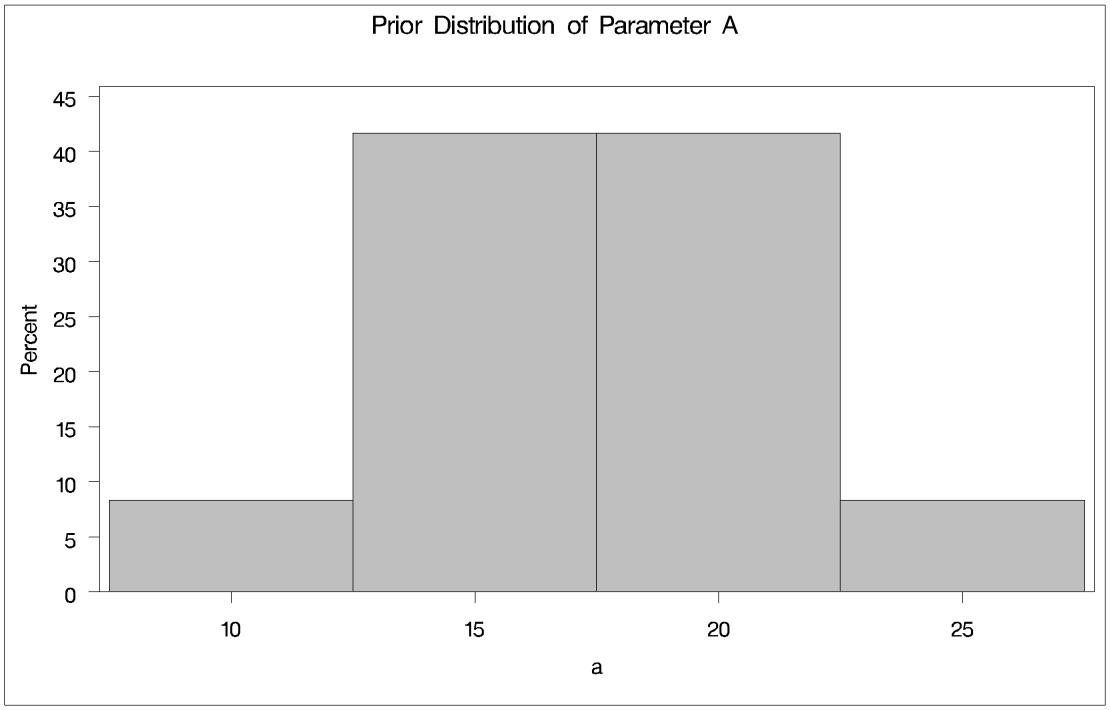


Figure 15.12. Prior Distribution of Parameter T

For the PRIOR data set created previously, the expected value of the coefficient of t is 2. The following SAS statements reestimate the parameters with prior specified for each one.

```
proc entropy data = prior outest=parm2;
   priors t 0(1) 2(3) 4(1)
      intercept -100(.5) -10(1.5) 0(2) 10(1.5) 100(0.5);
   model y = t;
   by by;
run;
```

The priors on the coefficient of t express a confident view of the value of the coefficient. The priors on INTERCEPT express a more diffuse view on the value of the intercept. The following PROC UNIVARIATE statement was used to compute summary statistics from the estimations:

```
proc univariate data=parm2 ;
  var t;
run;
```

The summary statistics for the distribution of the estimates of t are shown in Figure 15.13.

```
                         The UNIVARIATE Procedure
                               Variable:  t

                        Basic Statistical Measures

             Location                    Variability

         Mean     1.999872     Std Deviation            0.01404
         Median   2.000952     Variance               0.0001973
         Mode      .           Range                    0.06807
                               Interquartile Range      0.01641
```

Figure 15.13. Prior Information Monte Carlo Summary

The prior information improves the estimation of the coefficient of t dramatically. The downside of specifying priors comes when they are incorrect. For example, say the priors for this model were specified as

```
   priors t -2(1) 0(3) 2(1) ;
```

to indicate a prior centered on zero instead of two.

The resulting summary statistics shown in Figure 15.14 indicate how the estimation is biased away from the solution.

```
                         The UNIVARIATE Procedure
                               Variable:  t

                        Basic Statistical Measures

             Location                    Variability

         Mean     0.097888     Std Deviation            0.01193
         Median   0.096959     Variance               0.0001423
         Mode      .           Range                    0.06588
                               Interquartile Range      0.01607
```

Figure 15.14. Incorrect Prior Information Monte Carlo Summary

The more data available for estimation the less sensitive the parameters will be to the priors. If the number of observations in each sample was 50 instead of 10, then the summary statistics shown in Figure 15.15 are produced. The prior information is not supported by the data, so it is discounted.

```
                          The UNIVARIATE Procedure
                               Variable:  t

                          Basic Statistical Measures

                Location                    Variability

            Mean     0.838043     Std Deviation            0.01365
            Median   0.837825     Variance               0.0001864
            Mode      .           Range                    0.06555
                                  Interquartile Range      0.02008
```

Figure 15.15. Incorrect Prior Information with More Data

Pure Inverse Problems

A special case of systems of equations estimation is the pure inverse problem. A pure problem is one that contains an exact relationship between the dependent variable and the independent variables and does not have an error component. A pure inverse problem can be written as

$$\mathbf{y} = X\beta$$

where $\mathbf{y}$ is a n-dimensional vector of observations, X is a $n \times k$ matrix of regressors, and β is a k-dimensional vector of unknowns. Notice that there is no error term.

A classic example is Jaynes' (1963) dice problem. Given a six-sided die that can take on the values $x = 1, 2, 3, 4, 5, 6$ and the average outcome of the die $y = A$, compute the probabilities $\beta = (p_1, p_2, \cdots, p_6)'$ of rolling each number. This is two pieces of information to infer six values. The data points are the expected value of y and the sum of the probabilities is one. Given $E(y) = 4.0$, this problem is solved using the following SAS code:

```
data one;
   array x[6] ( 1 2 3 4 5 6 );
   y=4.0;
run;



proc entropy data=one pure;
   priors x1 0 1 x2 0 1 x3 0 1 x4 0 1 x5 0 1 x6 0 1;
   model y = x1-x6/ noint;
   restrict x1 + x2 +x3 +x4 + x5 + x6 =1;
run;
```

The probabilities are given in Figure 15.16.

```
                           The ENTROPY Procedure

                          GME Variable Estimates

                                          Information
         Variable         Estimate              Index    Label

         x1               0.101763             0.5254
         x2               0.122658             0.4630
         x3               0.147141             0.3974
         x4               0.175533             0.3298
         x5               0.208066             0.2622
         x6               0.244839             0.1970
         Restrict0        2.388082               .        x1 + x2 + x3 + x4
                                                         + x5 + x6  =  1
```

Figure 15.16. Jaynes' Dice Pure Inverse Problem

First Order Markov Process Estimation

A more useful inverse problem is the First Order Markov Process. Companies have a share of the marketplace where they do business. Generally customers for a specific market space can move from company to company. Graphically the movement of customers can be visualized as a flow diagram, as in Figure 15.17. The arrows represent movements of customers from one company to the next.

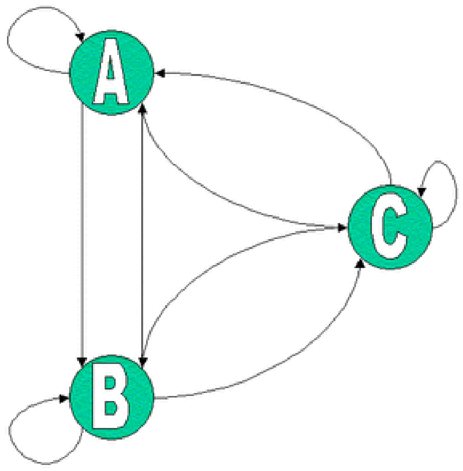


Figure 15.17. Markov Transition Diagram

You can model the probability that a customer moves from one company to another using a first-order Markov model. Mathematically that is

$$y_t = P y_{t-1}$$

where y_t is vector of k market shares at time t and P is a $k \times k$ matrix of unknown transition probabilities. p_{ij} represents the probability that a customer who is currently using company j at time $t-1$ will transition to using company i at time t. The diagonal elements then represent the probability that a customer stays with the current company. The columns in P sum to one.

Given market share information over time you can estimate the transition probabilities P. In order to estimate P using traditional methods you need at least k observations. If you have fewer than k transitions, you can use the ENTROPY procedure to estimate the probabilities.

Suppose you are studying the market share for four companies. If you wanted to estimate the transition probabilities for these four companies, you would need a time series with four observations of the shares. Assume the current transition probability matrix is as follows:

$$\begin{bmatrix} 0.7 & 0.4 & 0.0 & 0.1 \\ 0.1 & 0.5 & 0.4 & 0.0 \\ 0.0 & 0.1 & 0.6 & 0.0 \\ 0.2 & 0.0 & 0.0 & 0.9 \end{bmatrix}$$

The following SAS DATA step code is used to generate a series of market shares generated from this probability matrix. A transition is represented as the current period shares, y, and the previous period shares, x.

```
data m;
            /* Transition matrix */
  array p[4,4] (0.7 .4 .0 .1
                0.1 .5 .4 .0
                0.0 .1 .6 .0
                0.2 .0 .0 .9 ) ;
            /* Initial Market shares */
  array y[4] y1-y4 ( .4 .3 .2 .1 );
  array x[4] x1-x4;
  drop p1-p16 i;

  do i = 1 to 3;
    x[1] = y[1]; x[2] = y[2];
    x[3] = y[3]; x[4] = y[4];
    y[1] = p[1,1] * x1 + p[1,2] * x2 + p[1,3] * x3 + p[1,4] * x4;
    y[2] = p[2,1] * x1 + p[2,2] * x2 + p[2,3] * x3 + p[2,4] * x4;
    y[3] = p[3,1] * x1 + p[3,2] * x2 + p[3,3] * x3 + p[3,4] * x4;
    y[4] = p[4,1] * x1 + p[4,2] * x2 + p[4,3] * x3 + p[4,4] * x4;
    output;
  end;
run;
```

The following SAS statements estimate the transition matrix using only the first transition.

```
   proc entropy markov pure data=m(obs=1);
      model y1-y4 = x1-x4;
   run;
```

The MARKOV option implies NOINT for each model, and that the sum of the parameters in each column is one, and chooses support points of 0 and 1. This model can be expressed equivalently as

```
proc entropy pure data=m(obs=1) ;
   priors y1.x1 0 1 y1.x2 0 1 y1.x3 0 1 y1.x4 0 1;
   priors y2.x1 0 1 y2.x2 0 1 y2.x3 0 1 y2.x4 0 1;
   priors y3.x1 0 1 y3.x2 0 1 y3.x3 0 1 y3.x4 0 1;
   priors y4.x1 0 1 y4.x2 0 1 y4.x3 0 1 y4.x4 0 1;

   model y1 = x1-x4 / noint;
   model y2 = x1-x4 / noint;
   model y3 = x1-x4 / noint;
   model y4 = x1-x4 / noint;

   restrict y1.x1 + y2.x1 + y3.x1 + y4.x1 = 1;
   restrict y1.x2 + y2.x2 + y3.x2 + y4.x2 = 1;
   restrict y1.x3 + y2.x3 + y3.x3 + y4.x3 = 1;
   restrict y1.x4 + y2.x4 + y3.x4 + y4.x4 = 1;
run;
```

The transition matrix is given in Figure 15.18.

```
                          The ENTROPY Procedure

                         GME Variable Estimates

                                               Information
               Variable        Estimate              Index

               y1.x1           0.463407             0.0039
               y1.x2            0.41055             0.0232
               y1.x3           0.356272             0.0605
               y1.x4           0.302163             0.1161
               y2.x1           0.272755             0.1546
               y2.x2           0.271459             0.1564
               y2.x3           0.267252             0.1625
               y2.x4           0.260084             0.1731
               y3.x1           0.119926             0.4709
               y3.x2           0.148481             0.3940
               y3.x3           0.180224             0.3194
               y3.x4           0.214394             0.2502
               y4.x1           0.143903             0.4056
               y4.x2           0.169504             0.3434
               y4.x3           0.196252             0.2856
               y4.x4           0.223364             0.2337
```

Figure 15.18. Estimate of P using One Transition

Note that P varies greatly from the true solution.

If two transitions are used instead (OBS=2), the resulting transition matrix is shown in Figure 15.19.

```
proc entropy markov pure data=m(obs=2);
   model y1-y4 = x1-x4;
run;
```

```
                     The ENTROPY Procedure

                    GME Variable Estimates

                                           Information
          Variable        Estimate               Index

          y1.x1           0.721012              0.1459
          y1.x2           0.355703              0.0609
          y1.x3           0.026095              0.8256
          y1.x4           0.096654              0.5417
          y2.x1           0.083987              0.5839
          y2.x2            0.53886              0.0044
          y2.x3           0.373668              0.0466
          y2.x4           0.000133              0.9981
          y3.x1           0.000062              0.9990
          y3.x2           0.099848              0.5315
          y3.x3           0.600104              0.0291
          y3.x4           7.871E-8              1.0000
          y4.x1           0.194938              0.2883
          y4.x2            0.00559              0.9501
          y4.x3           0.000133              0.9981
          y4.x4           0.903214              0.5413
```

Figure 15.19. Estimate of P using Two Transitions

This transition matrix is much closer to the actual transition matrix.

If in addition to the transitions, you had other information about the transition matrix, such as your own company's transition values, that information can be added as restrictions to the parameter estimates. For noisy data, the PURE option should be dropped. Note that this example had six zero probabilities in the transition matrix, the accurate estimation of transition matrices with fewer zero probabilities generally requires more transition observations.

Analyzing Multinomial Response Data

Multinomial discrete choice models suffer the same problems with collinearity of the regressors and small sample sizes as linear models. Unordered multinomial discrete choice models can be estimated using a variant of GME for discrete models called GME-D.

Consider the model shown in Golan, Judge, and Perloff (1996). In this model there are five occupational categories and the categories where considered a function of four individual characteristics plus an intercept. The sample contains 337 individuals.

```
data kpdata;
   input job x1 x2 x3 x4;
datalines;
0 1 3 11 1
0 1 14 12 1
0 1 44 12 1
0 1 18 12 1

  ...

4 1 12 16 1
;
run;
```

The dependent variable in this data, job, takes on values 1 through 5. Support points are used only for the error terms so priors are specified on the MODEL statement.

```
proc entropy data = kpdata gmed;
   model job = x1 x2 x3 x4 / noint
      priors=( -.1 -0.0666 -0.0333 0 0.0333 0.0666 .1 );
run;
```

```
                           The ENTROPY Procedure

                          GME-D Variable Estimates

                                  Approx                    Approx
    Variable        Estimate     Std Err     t Value     Pr > |t|

    x1_0                   0      6.5099        0.00       1.0000
    x2_0                   0    0.000875        0.00       1.0000
    x3_0                   0      0.0212        0.00       1.0000
    x4_0                   0      1.9930        0.00       1.0000
    x1_1            2.349425      6.2302        0.38       0.7063
    x2_1            -0.00518    0.000832       -6.23       <.0001
    x3_1            -0.19324      0.0200       -9.65       <.0001
    x4_1            0.771261      2.0005        0.39       0.7001
    x1_2            0.637759      6.0423        0.11       0.9160
    x2_2            0.017179    0.000811       21.17       <.0001
    x3_2            -0.01251      0.0192       -0.65       0.5150
    x4_2            0.042352      1.9000        0.02       0.9822
    x1_3            -4.63595      6.6816       -0.69       0.4883
    x2_3            0.025678    0.000848       30.27       <.0001
    x3_3            0.248569      0.0207       12.01       <.0001
    x4_3            1.213053      2.1544        0.56       0.5738
    x1_4            -9.07911      6.4745       -1.40       0.1618
    x2_4            0.024826    0.000828       30.00       <.0001
    x3_4            0.634031      0.0202       31.43       <.0001
    x4_4            1.190452      1.9838        0.60       0.5489
```

Figure 15.20. Estimate of Jobs Model Using GME-D

Note there are five estimates of the parameters produced for each regressor, one for each choice. The first choice is restricted to zero for normalization purposes.

Syntax

The following statements can be used with the ENTROPY procedure:

PROC ENTROPY *options*;
  BOUNDS *bound1 [, bound2, . . .]* ;
  BY *variable [variable . . .]* ;
  ID *variable [variable . . .]* ;
  MODEL *variable = variable [variable] . . . [/ options] ;*
  PRIORS *variable [support points] variable [value] . . .* ;
  RESTRICT *restriction1 [, restriction2 . . .]* ;
  TEST *[* **"name"** *] test1 [, test2 . . .] [/ options]* ;
  WEIGHT *variable*;

Functional Summary

The statements and options in the ENTROPY procedure are summarized in the following table.

Description	**Statement**	**Option**
Data Set Options		
specify the input data set for the variables	ENTROPY	DATA=
specify the input data set for support points and priors	ENTROPY	PDATA=
specify the output data set for residual, predicted, or actual values	ENTROPY	OUT=
specify the output data set for the support points and priors	ENTROPY	OUTP=
write the covariance matrix of the estimates to OUTEST= data set	ENTROPY	OUTCOV
write the parameter estimates to a data set	ENTROPY	OUTEST=
write the Lagrange multiplier estimates to a data set	ENTROPY	OUTL=
write the covariance matrix of the equation errors to a data set	ENTROPY	OUTS=
write the **S** matrix used in the objective function definition to a data set	ENTROPY	OUTSUSED=
read the covariance matrix of the equation errors	ENTROPY	SDATA=
Printing Options		
print collinearity diagnostics	ENTROPY	COLLIN
Options to Control Iteration Output		

Description	**Statement**	**Option**
print a summary iteration listing	ENTROPY	ITPRINT
Options to Control the Minimization Process		
specify the convergence criteria	ENTROPY	CONVERGE=
specify the maximum number of iterations allowed	ENTROPY	MAXITER=
specify the maximum number of subiterations allowed	ENTROPY	MAXSUBITER=
select the iterative minimization method to use	ENTROPY	METHOD=
specify a weight variable	WEIGHT	
General Printing Control Options		
suppress the normal printed output	ENTROPY	NOPRINT
Statements That Declare Variables		
subset the data set with *by* variables	BY	
specify identifying variables	ID	
General ENTROPY Statement Options		
specify seemingly unrelated regression	ENTROPY	SUR
specify iterated seemingly unrelated regression	ENTROPY	ITSUR
specify data-constrained generalized maximum entropy	ENTROPY	GME
specify normed moment generalized maximum entropy	ENTROPY	GMENM
specify the denominator for computing variances and covariances	ENTROPY	VARDEF=
General TEST Statement Options		
specify that a Wald test be computed	TEST	WALD
specify that a Lagrange multiplier test be computed	TEST	LM
specify that a likelihood ratio test be computed	TEST	LR
requests all three types of tests	TEST	ALL

PROC ENTROPY Statement

PROC ENTROPY *options;*

The following options can be specified in the PROC ENTROPY statement.

General Options

COLLIN
requests that the collinearity analysis of the $X'X$ matrix be performed.

GME|GCE
requests generalized maximum entropy or generalized cross entropy.

GMENM|GCENM
requests normed moment maximum entropy.

MARKOV
specifies that the model is a Markov model.

PURE
specifies a regression without an error term.

SUR | ITSUR
specifies seemingly unrelated regression estimation.

VARDEF=N | WGT | DF | WDF
specifies the denominator to be used in computing variances and covariances. VARDEF=N specifies that the number of nonmissing observations be used. VARDEF=WGT specifies that the sum of the weights be used. VARDEF=DF specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used. VARDEF=WDF specifies that the sum of the weights minus the model degrees of freedom be used. The default is VARDEF=DF.

Data Set Options

DATA= *SAS-data-set*
specifies the input data set. Values for the variables in the model are read from this data set.

COVBEST= CROSS | GME | GMENM *SAS-data-set*
specifies the method for producing the covariance matrix for output and for standard error calculations. The default is GME or GMENM, depending on which method is selected for estimation. If a GME estimation is performed, COVBEST=GMENM or COVBEST=GME results in GME being used to estimate the covariance matrix. The same is true for GMENM.

OUT= *SAS-data-set*
names the SAS data set to contain the residuals, from each estimation.

OUTCOV
COVOUT
writes the covariance matrix of the estimates to the OUTEST= data set in addition to the parameter estimates. The OUTCOV option is applicable only if the OUTEST= option is also specified.

OUTEST= *SAS-data-set*
names the SAS data set to contain the parameter estimates and optionally the covariance of the estimates.

OUTL= *SAS-data-set*
names the SAS data set to contain the estimated Lagrange multipliers for the models.

OUTP= *SAS-data-set*
names the SAS data set to contain the support points and estimated probabilities.

OUTS= *SAS-data-set*
names the SAS data set to contain the estimated covariance matrix of the equation errors. This is the covariance of the residuals computed from the parameter estimates.

OUTSUSED= *SAS-data-set*
names the SAS data set to contain the S matrix used in the objective function definition. The OUTSUSED= data set is the same as the OUTS= data set for the methods that iterate the S matrix.

SDATA= *SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The matrix read from the SDATA= data set is used for the equation covariance matrix (S matrix) in the estimation. (The SDATA= S matrix is used to provide only the initial estimate of *S* for the methods that iterate the S matrix.)

Printing Options

ITPRINT
prints the parameter estimates, objective function value, and convergence criteria at each iteration.

NOPRINT
suppresses the normal printed output but does not suppress error listings. Using any other print option turns the NOPRINT option off.

Options to Control the Minimization Process

The following options may be helpful when you experience a convergence problem. In addition to the options listed here, in the PROC ENTROPY statement you can also use any option supported by the NLO subsystem. See Chapter 10, “Nonlinear Optimization Methods,” for more details.

CONVERGE= *value*
GCONV= *value*
specifies the convergence criteria for *S*-iterated methods. The convergence measure must be less than *value* before convergence is assumed. The default value is CONVERGE=.001.

DUAL | PRIMAL
specifies whether the optimization problem is solved using the dual or primal form. The dual formalization is the default.

MAXITER= *n*
specifies the maximum number of iterations allowed. The default is MAXITER=100.

MAXSUBITER= *n*
specifies the maximum number of subiterations allowed for an iteration. The MAXSUBITER= option limits the number of step halvings. The default is MAXSUBITER=30.

METHOD= TR | NEWRAP | NRR | QN
TECHNIQUE= TR | NEWRAP | NRR | QN
TECH= TR | NEWRAP | NRR | QN
specifies the iterative minimization method to use. METHOD=TR specifies the Trust Region method, METHOD=NEWRAP specifies the Newton Raphson method, METHOD=NRR specifies the Newton Raphson-Ridge method, and METHOD=QN specifies the Quasi-Newton method. See Chapter 10, "Nonlinear Optimization Methods," for more details on optimization methods. The default is METHOD=QN for the dual estimation and METHOD=NEWRAP for the primal estimation.

BOUNDS Statement

BOUNDS *bound1 [, bound2 ...] ;*

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the ENTROPY Procedure. You can specify any number of BOUNDS statements.

Each *bound* is composed of variables and constants and inequality operators:

item operator item [operator item [operator item . . .]]

Each *item* is a constant, the name of an regressor variable, or a list of regressor names. Each *operator* is '<', '>', '<=', or '>='.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See the "RESTRICT Statement" section on page 759 for more information on the computational details of estimation with inequality restrictions.

Lagrange multipliers are reported for all the active boundary constraints. In the printed output and in the OUTEST= data set, the Lagrange multiplier estimates are identified with the names BOUND1, BOUND2, and so forth. The probability of the Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive or nonbinding bounds have no effect on the estimation results and are not noted in the output. To give the constraints more descriptive names, use the RESTRICT statement instead of the BOUNDS statement.

The following BOUNDS statement constrains the estimates of the coefficients of WAGE and TARGET and the 10 coefficients of $x1$ through $x10$ to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < wage target x1-x10 < 1;
```

The following is an example of the use of the BOUNDS statement:

```
proc entropy data=zero ;
   bounds .1 <= x1 <= 100,
          0 <= x2 <=  25.6,
          0 <= x3 <=   5;

   model y = x1 x2 x3;
run;
```

The parameter estimates from this run are shown in Figure 15.21.

```
                       GME-NM Variable Estimates

                         Approx             Approx
Variable     Estimate   Std Err t Value   Pr > |t| Label

x1                0.1         0      .         .
x2              4E-16         0      .         .
x3           5.12E-18         0      .         .
Intercept    -0.00432 3.406E-6 -1269.3    <.0001
Bound0       1.257309    9130.3    0.00    0.9999  0.1  <=  x1
Bound2       0.009384         0      .         .   0  <=  x2
Bound4       0.000032         0      .         .   0  <=  x3
```

Figure 15.21. Output from Bounded Estimation

BY Statement

BY *variables;*

A BY statement is used to obtain separate estimates for observations in groups defined by the BY variables. To save parameter estimates for each BY group, use the OUTEST= option.

ID Statement

ID *variables;*

The ID statement specifies variables to identify observations in error messages or other listings and in the OUT= data set. The ID variables are normally SAS date or datetime variables. If more than one ID variable is used, the first variable is used to identify the observations; the remaining variables are added to the OUT= data set.

Model Statement

MODEL *dependent = regressors / options ;*

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. If no independent variables are specified in the MODEL statement, only the mean (intercept) is estimated.

The following options can be used in the MODEL statement after a slash (/).

PRIORS=(support (prior) . . . **)**

specifies the support points and prior weights on the residuals for this equation. The default is the following five support values

$$-10 * value, -value, 0, value, 10 * value$$

where $value$ is computed as

$$value = (max(y) - \bar{y}) * multiplier$$

for GME and

$$value = (max(y) - \bar{y}) * multiplier * nobs * max(X) * 0.1$$

for GME-NM. The $multiplier$ depends on the MULTIPLIER= option. The MULTIPLIER= option defaults to 2 for unrestricted models and to 4 for restricted models. The prior probabilities default to

$$0.0005, .333, .333, .333, 0.0005$$

The probabilities and supports are selected so that hypothesis tests can be performed without adding significant bias to the estimation. These values are ad hoc.

NOINT

suppresses the intercept parameter.

PRIORS Statement

PRIORS *variable [support points [(priors)]] [variable [[support points [(priors)]]] ... ;*

The PRIORS statement specifies the support points and priors for the coefficients on the variables.

Support points for coefficients default to five points, determined as follows:

$$-2 * value, -value, 0, value, 2 * value$$

where $value$ is computed as

$$value = (\|mean\| + 3 * stderr) * multiplier$$

where the $mean$ and the $stderr$ are obtained from OLS, and the $multiplier$ depends on the MULTIPLIER= option. The MULTIPLIER= option defaults to 2 for unrestricted models and to 4 for restricted models. The prior probabilities for each support point default to the uniform distribution.

The number of support points must be at least two. If priors are specified, they must be positive and there must be the same number of priors as there are support points. Priors and support points can also be specified through the PDATA= data set.

RESTRICT Statement

RESTRICT *restriction1 [, restriction2 ...] ;*

The RESTRICT statement is used to impose linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an optional name, followed by an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

["name"] expression operator expression

The optional *"name"* is a string used to identify the restriction in the printed output and in the OUTEST= data set. The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional, as in the TEST statement it defaults to $= 0$.

Restriction expressions can be composed of variable names, times ($*$), and plus ($+$) operators, and constants. Variable names in test expressions must be among the variables estimated by the model. The restriction expressions must be a linear function of the variables.

The following is an example of the use of the RESTRICT statement:

```
proc entropy data=one;
   restrict y1.x1*2 <= x2 + y2.x1;
   model y1 = x1 x2;
   model y2 = x1 x3;
run;
```

This example illustrates the use of compound names, y1.x1, to specify coefficients of specific equations.

TEST Statement

TEST *["name"] test1 [, test2 ...] [,/ options] ;*

The TEST statement performs tests of linear hypotheses on the model parameters.

The TEST statement applies only to parameters estimated in the model. You can specify any number of TEST statements.

Each test is written as an expression optionally followed by an equal sign (=) and a second expression:

[expression] [= expression]

Test expressions can be composed of variable names, times ($*$), plus ($+$), and minus ($-$) operators, and constants. Variables named in test expressions must be among the variables estimated by the model.

If you specify only one expression in a test, that expression is tested against zero. For example, the following two TEST statements are equivalent:

```
test a + b;

test a + b = 0;
```

When you specify multiple tests on the same TEST statement, a joint test is performed. For example, the following TEST statement tests the joint hypothesis that both of the coefficients on a and b are equal to zero.

```
test a, b;
```

To perform separate tests rather than a joint test, use separate TEST statements. For example, the following TEST statements test the two separate hypotheses that a is equal to zero and that b is equal to zero.

```
test a;
test b;
```

You can use the following options in the TEST statement.

WALD
specifies that a Wald test be computed. WALD is the default.

LM
RAO
LAGRANGE
specifies that a Lagrange multiplier test be computed.

LR
LIKE
specifies that a pseudo likelihood ratio test be computed.

ALL
requests all three types of tests.

OUT=
specifies the name of an output SAS data set that contains the test results. The format of the OUT= data set produced by the TEST statement is similar to that of the OUTEST= data set produced by the ENTROPY statement.

WEIGHT Statement

WEIGHT *variable;*

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters.

If the weight of an observation is nonpositive, that observation is not used for the estimation. *variable* must be a numeric variable in the input data set. The regressors and the dependent variables are multiplied by the square root of the weight to form the X matrix and the left-hand side of the regression.

Details

Shannon's measure of entropy for a distribution is given by:

$$\begin{aligned} &\textbf{Maximize} && -\sum_{i=1}^{n} p_i \ln(p_i) \\ &\textit{subject to:} && \sum_{i=1}^{n} p_i = 1 \end{aligned}$$

where p_i is the probability associated with the *i*th support point. Properties that characterize the entropy measure are set forth by Kapur and Kesavan (1992).

The idea is to maximize the entropy of the distribution with respect to the probabilities p_i subject to constraints that reflect any other known information about the distribution (Jaynes 1957). This measure, in the absence of additional information, reaches a maximum when the probabilities are uniform. A distribution other than the uniform distribution arises from information already known.

Generalized Maximum Entropy

Reparameterization of the errors in a regression equation is the process of specifying a support for the errors, observation by observation. If a two-point support is used, the error for the *t*th observation is reparameterized by setting $e_t = w_{t1}\, v_{t1} + w_{t2}\, v_{t2}$ where v_{t1} and v_{t2} are the upper and lower bounds for the *t*th error e_t, and w_{t1} and w_{t2} represent the weight associated with the point v_{t1} and v_{t2}. The error distribution is usually chosen to be symmetric, centered about zero, and the same across observations so that $v_{t1} = -v_{t2} = R$, where *R* is the support value chosen for the problem (Golan, Judge, and Miller (1996)).

The Generalized Maximum Entropy (GME) formulation was proposed for the ill-posed or underdetermined case where there is insufficient data to estimate the system with traditional methods. β is reparameterized by defining a support for β (and a set of weights in the cross entropy case), which define a prior distribution for β.

In the simplest case, each β_k is reparameterized as $\beta_k \;=\; p_{k1}\, z_{k1} \;+\; p_{k2}\, z_{k2}$ where p_{k1} and p_{k2} represent the probabilities ranging from [0,1] for each β, and z_{k1} and z_{k2} represent the lower and upper bounds placed on β_k. The support points, z_{k1} and z_{k2}, are usually distributed symmetrically around the most likely value for β_k based on some prior knowledge.

With these reparameterizations, the GME estimation problem is

$$\begin{array}{ll}
\textbf{Maximize} & H(p,w) \;=\; -p' \ln(p) \;-\; w' \ln(w) \\
\textit{subject to:} & y \;=\; X\, Z\, p \;+\; V\, w \\
 & 1_K \;=\; (I_K \,\otimes\, 1_L')\, p \\
 & 1_T \;=\; (I_T \,\otimes\, 1_L')\, w
\end{array}$$

where

$$\beta \;=\; Z\,p \;=\; \left[\begin{array}{cccccccccc}
z_{11} & \cdots & z_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & z_{12} & \cdots & z_{L2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & z_{1K} & \cdots & z_{LK}
\end{array}\right]
\left[\begin{array}{c}
p_{11} \\ \vdots \\ p_{L1} \\ p_{12} \\ \vdots \\ p_{L2} \\ \vdots \\ p_{1K} \\ \vdots \\ p_{LK}
\end{array}\right]$$

$$e \;=\; V\,w \;=\; \left[\begin{array}{cccccccccc}
v_{11} & \cdots & v_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & v_{12} & \cdots & v_{L2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & v_{1T} & \cdots & v_{LT}
\end{array}\right]
\left[\begin{array}{c}
w_{11} \\ \vdots \\ w_{L1} \\ w_{12} \\ \vdots \\ w_{L2} \\ \vdots \\ w_{1T} \\ \vdots \\ w_{LT}
\end{array}\right]$$

y denotes the *T* column vector of observations of the dependent variables; *X* denotes the $(T \times K)$ matrix of observations of the independent variables; *p* denotes the *LK* column vector of weights associated with the points in *Z*; *w* denotes the *LT* column vector of weights associated with the points in *V*; 1_K, 1_L, and 1_T are *K*-, *L*-, and

T-dimensional column vectors, respectively, of ones; and I_K and I_T are $(K \times K)$ and $(T \times T)$ dimensional identity matrices.

These equations can be rewritten using set notation.

$$\textbf{Maximize} \quad H(p,w) \;=\; -\sum_{l=1}^{L}\sum_{k=1}^{K} p_{kl}\,\ln(p_{kl}) \;-\; \sum_{l=1}^{L}\sum_{t=1}^{T} w_{tl}\,\ln(w_{tl})$$

$$\textit{subject to:} \quad y_t \;=\; \sum_{l=1}^{L}\left[\sum_{k=1}^{K}\left(X_{kt}\,Z_{kl}\,p_{kl}\right) \;+\; V_{tl}\,w_{tl}\right]$$

$$\sum_{l=1}^{L} p_{kl} \;=\; 1 \;\text{ and }\; \sum_{l=1}^{L} w_{tl} \;=\; 1$$

The subscript *l* denotes the support point (l=1,2,..,L), *k* denotes the parameter (k=1,2,..K), and *t* denotes the observation (t=1,2,..,T).

The GME objective is strictly concave; therefore, a unique solution exists. The optimal estimated probabilities, *p* and *w*, and the prior supports, *Z* and *V*, can be used to form the point estimates of the unknown parameters, β, and the unknown errors, *e* .

Generalized Cross Entropy

Kullback and Leibler's (1951) cross entropy measures the "discrepancy" between one distribution and another. Cross entropy is called a measure of discrepancy rather than distance because it does not satisfy some of the properties one would expect of a distance measure. (See Kapur and Kesavan [1992] for a discussion of cross entropy as a measure of discrepancy.) Mathematically, cross entropy is written as

$$\textbf{Minimize} \quad \sum_{i=1}^{n} p_i \ln(\,p_i \,/\, q_i)$$

$$\textit{subject to:} \quad \sum_{i=1}^{n} p_i \;=\; 1,$$

where q_i is the probability associated with the *i*th point in the distribution from which the discrepancy is measured. The q_i (in conjunction with the support) are often referred to as the prior distribution. The measure is nonnegative and is equal to zero when p_i equals q_i. The properties of the cross entropy measure are examined by Kapur and Kesavan (1992).

The Principle of Minimum Cross Entropy (Kullback 1959; Good 1963) states that one should choose probabilities that are as close as possible to the prior probabilities.

That is, out of all probability distributions satisfying a given set of constraints reflecting known information about the distribution, choose the distribution that is closest (as measured by $p(\ln(p) - \ln(q))$refgce1) to the prior distribution. When the prior distribution is uniform, maximum entropy and minimum cross entropy produce the same results (Kapur and Kesavan 1992), with higher values for entropy corresponding exactly with the lower values for cross entropy.

If the prior distributions are nonuniform, the problem can be stated as a Generalized Cross Entropy (GCE) formulation. The cross entropy terminology specifies weights, q_i and u_i, for the points Z and V, respectively. Given informative prior distributions on Z and V, the GCE problem is

$$\begin{aligned} \textbf{Minimize} \quad I(p,q,w,u) \;&=\; \frac{p'\,\ln(p/q)}{w'\,\ln(w/u)} \\ \textit{subject to:} \quad y \;&=\; X\,Z\,p \;+\; V\,w \\ 1_K \;&=\; (I_K \otimes 1'_L)\,p \\ 1_T \;&=\; (I_T \otimes 1'_L)\,w \end{aligned}$$

where *y* denotes the *T* column vector of observations of the dependent variables; *X* denotes the $(T \times K)$ matrix of observations of the independent variables; *q* and *p* denote *LK* column vectors of prior and posterior weights, respectively, associated with the points in *Z*; *u* and *w* denote the *LT* column vectors of prior and posterior weights, respectively, associated with the points in *V*; 1_K, 1_L, and 1_T are *K*- , *L*-, and *T*-dimensional column vectors, respectively, of ones; and I_K and I_T are ($K \times K$) and ($T \times T$) dimensional identity matrices.

The optimization problem can be rewritten using set notation.

$$\textbf{Minimize} \quad I(p,q,w,u) \;=\; \sum_{l=1}^{L}\sum_{k=1}^{K} p_{kl}\,\ln(p_{kl}/q_{kl}) \;+\; \sum_{l=1}^{L}\sum_{t=1}^{T} w_{tl}\,\ln(w_{tl}/u_{tl})$$

$$\textit{subject to:} \quad y_t \;=\; \sum_{l=1}^{L}\left[\sum_{k=1}^{K}\left(X_{kt}\,Z_{kl}\,p_{kl}\right) \;+\; V_{tl}\,w_{tl}\right]$$

$$\sum_{l=1}^{L} p_{kl} \;=\; 1 \;\text{ and }\; \sum_{l=1}^{L} w_{tl} \;=\; 1$$

The subscript *l* denotes the support point (l=1,2,..,L), *k* denotes the parameter (k=1,2,..K), and *t* denotes the observation (t=1,2,..,T).

The objective function is strictly convex; therefore, there is a unique global minimum for the problem (Golan, Judge, and Miller 1996). The optimal estimated weights, *p* and *w*, and the prior supports, *Z* and *V*, can be used to form the point estimates of the unknown parameters, β, and the unknown errors, *e* using

$$\beta = Z p = \begin{bmatrix} z_{11} & \cdots & z_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_{12} & \cdots & z_{L2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_{1K} & \cdots & z_{LK} \end{bmatrix} \begin{bmatrix} p_{11} \\ \vdots \\ p_{L1} \\ p_{12} \\ \vdots \\ p_{L2} \\ \vdots \\ p_{1K} \\ \vdots \\ p_{LK} \end{bmatrix}$$

$$e = V w = \begin{bmatrix} v_{11} & \cdots & v_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{12} & \cdots & v_{L2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & v_{1T} & \cdots & v_{LT} \end{bmatrix} \begin{bmatrix} w_{11} \\ \vdots \\ w_{L1} \\ w_{12} \\ \vdots \\ w_{L2} \\ \vdots \\ w_{1T} \\ \vdots \\ w_{LT} \end{bmatrix}$$

Computational Details

This constrained estimation problem can either be solved directly (Primal) or by using the dual form. Either way, it is prudent to factor out one probability for each parameter and each observation as the sum of the other probabilities. This factoring reduces the computational complexity significantly. If the primal formalization is used and two support points are used for the parameters and the errors, the resulting GME problem is O($(nparms + nobs)^3$). For the dual form the problem is O($(nobs)^3$). Therefore for large data sets, GME-NM should be used instead of GME.

Normed Moment Generalized Maximum Entropy

The default estimation technique is Normed Moment Generalized Maximum Entropy (GME-NM). This is simply GME with the data constraints modified by multiplying both sides by X'. GME-NM then becomes

$$\textbf{Maximize} \quad H(p, w) = -p' \ln(p) - w' \ln(w)$$

$$\begin{aligned} \textit{subject to:} \quad X'y &= X'X\,Z\,p + X'V\,w \\ 1_K &= (I_K \otimes 1'_L)\,p \\ 1_T &= (I_T \otimes 1'_L)\,w \end{aligned}$$

There is also the Cross Entropy version of GME-NM, which has the same form as GCE but with the normed constraints.

GME versus GME-NM

GME-NM is more computationally attractive than GME for large data sets because the computational complexity of estimation problem primarily depends on the number of parameters and not on the number of observations. GME-NM is based on the first moment of the data whereas GME is based on the data itself. If the distribution of the residuals is well defined by its first moment, then GME-NM is a good choice. So if the residuals are normally distributed or exponentially distributed, then GME-NM should be used. On the other hand if the distribution is Cauchy, lognormal, or other distribution, where the first moment does not describe the distribution, then use GME. See Example 15.1 for an illustration of this point.

Maximum Entropy-based Seemingly Unrelated Regression

In a multivariate regression model, the errors in different equations may be correlated. In this case, the efficiency of the estimation may be improved by taking these cross-equation correlations into account. Seemingly unrelated regression (SUR), also called joint generalized least squares (JGLS) or Zellner estimation, is a generalization of OLS for multi-equation systems.

Like SUR in the least squares setting, the Maximum Entropy SUR (GME-SUR) method assumes that all the regressors are independent variables, and uses the correlations among the errors in different equations to improve the regression estimates. The GME-SUR method requires an initial entropy regression to compute residuals. The entropy residuals are used to estimate the cross-equation covariance matrix.

In the iterative GME-SUR (ITGME-SUR) case, the preceding process is repeated using the residuals from the GME-SUR estimation to estimate a new cross-equation covariance matrix. ITGME-SUR method alternates between estimating the system coefficients and estimating the cross-equation covariance matrix until the estimated coefficients and covariance matrix converge.

The estimation problem becomes the Generalized Maximum Entropy system adapted for multi-equations.

$$\begin{aligned} \textbf{Maximize} \quad H(p,w) &= -p'\ln(p) - w'\ln(w) \\ \textit{subject to:} \quad y &= X\,Z\,p + V\,w \\ 1_{KM} &= (I_{KM} \otimes 1'_L)\,p \\ 1_{MT} &= (I_{MT} \otimes 1'_L)\,w \end{aligned}$$

where

$$\beta \ = \ Z\,p$$

$$Z \ = \ \begin{bmatrix} z_{11}^{1} & \cdots & z_{L1}^{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & z_{11}^{K} & \cdots & z_{L1}^{K} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_{1M}^{1} & \cdots & z_{LM}^{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_{1M}^{K} & \cdots & z_{LM}^{K} \end{bmatrix}$$

$$p \ = \ \left[\ p_{11}^{1} \ \cdot \ p_{L1}^{1} \ \cdot \ p_{11}^{K} \ \cdot \ p_{L1}^{K} \ \cdot \ p_{1M}^{1} \ \cdot \ p_{LM}^{1} \ \cdot \ p_{1M}^{K} \ \cdot \ p_{LM}^{K}\ \right]'$$

$$e \ = \ V\,w$$

$$V \ = \ \begin{bmatrix} v_{11}^{1} & \cdots & v_{11}^{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v_{1T}^{1} & \cdots & v_{1T}^{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v_{M1}^{1} & \cdots & v_{M1}^{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v_{MT}^{1} & \cdots & v_{MT}^{L} \end{bmatrix}$$

$$w \ = \ \left[\ w_{11}^{1} \ \cdot \ w_{11}^{L} \ \cdot \ w_{1T}^{1} \ \cdot \ w_{1T}^{L} \ \cdot \ w_{M1}^{1} \ \cdot \ w_{M1}^{L} \ \cdot \ w_{MT}^{1} \ \cdot \ w_{MT}^{L}\ \right]'$$

y denotes the *MT* column vector of observations of the dependent variables; *X* denotes the (*MT* x *KM*) matrix of observations for the independent variables; *p* denotes the *LKM* column vector of weights associated with the points in *Z*; *w* denotes the *LMT* column vector of weights associated with the points in *V*; 1_L, 1_{KM}, and 1_{MT} are *L*-, *KM*-, and *MT*-dimensional column vectors, respectively, of ones; and I_{KM} and I_{MT} are (*KM* x *KM*) and (*MT* x *MT*) dimensional identity matrices. The subscript *l* denotes the support point (l=1,2,..,L), *k* denotes the parameter (k=1,2,..K), *m* denotes the equation (m=1,2,...,M), and *t* denotes the observation (t=1,2,..,T).

Using this notation, the Maximum Entropy problem that is analogous to the OLS problem used as the initial step of the traditional SUR approach is:

Maximize $H(p,w) \ = \ -p' \ln(p) \ - \ w' \ln(w)$

subject to: $(y \ - \ X\,Z\,p) \ = \sqrt{\Sigma}\,V\,w$

$$1_{KM} \ = \ (I_{KM} \otimes 1_L')\,p$$

$$1_{MT} \ = \ (I_{MT} \otimes 1_L')\,w$$

The results are GME-SUR estimates with independent errors, the analog of OLS. The covariance matrix $\hat{\Sigma}$ is computed based on the residual of the equations, $Vw = e$. An $L'L$ factorization of the $\hat{\Sigma}$ is used to compute the square root of the matrix.

After solving this problem, these entropy-based estimates are analogous to Aitken 2-step. For iterative GME-SUR, the covariance matrix of the errors is recomputed, and a new $\hat{\Sigma}$ is computed and factored. As in traditional ITSUR, this process repeats until the covariance matrix and the parameter estimates converge.

The estimation of the parameters for the normed-moment version of SUR, GME-SUR-NM, uses an identical process. The constraints for the normed-moment version of SUR, GME-SUR-NM, is defined as:

$$X'y \;=\; X'(\mathbf{S}^{-1}\otimes\mathbf{I})X\,Z\,p \;+\; X'(\mathbf{S}^{-1}\otimes\mathbf{I})V\,w$$

The estimation of the parameters for GME-SUR-NM uses an identical process as outlined previously for GME-SUR.

Generalized Maximum Entropy for Multinomial Discrete Choice Models

Multinomial discrete choice models take the form of an experiment consisting of n trials. On each trial, one of k alternatives is observed. If y_{ij} is the random variable that takes on the value 1 when alternative j is selected for the ith trial and 0 otherwise, then the probability that y_{ij} is 1 conditional on a vector of regressors X_i and unknown parameter vector β_j is

$$\Pr(y_{ij} = 1|X_i, \beta_j) = G(X_i'\beta_j)$$

where $G()$ is a link function. For noisy data the model becomes

$$y_{ij} = G(X_i'\beta_j) + \epsilon_{ij} = p_{ij} + \epsilon_{ij}$$

The standard maximum likelihood approach for multinomial logit is equivalent to the maximum entropy solution for discrete choice models. The generalized maximum entropy approach avoids an assumption of the form of the link function $G()$.

The generalized maximum entropy for discrete choice models (GME-D) is written in primal form as

$$\begin{array}{lll}
\textbf{Maximize} & H(p,w) & = -p'\ \ln(p)\ -\ w'\ \ln(w) \\
\textit{subject to:} & (I_j \otimes X'y) & = (I_j \otimes X')p\ +\ (I_j \otimes X')V\,w \\
 & \sum_j^k p_{ij} & = 1 \quad \text{for } i = 1 \text{ to } N \\
 & \sum_m^L w_{ijm} & = 1 \quad \text{for } i = 1 \text{ to } N \text{ and } j = 1 \text{ to } k
\end{array}$$

The parameters β_j are the Lagrange multipliers of the constraints. The covariance matrix of the parameter estimates is computed as the inverse of the Hessian of the dual form of the objective function.

Information Measures

Normalized entropy, NE, measures the relative informational content of both the signal and noise components through *p* and *w*, respectively (Golan, Judge, and Miller 1996). Let *S* denote the normalized entropy of the signal, $X\beta$, defined as

$$S(\tilde{p}) = \frac{-\tilde{p}' \ln(\tilde{p})}{-q' \ln(q)}$$

where $S(\tilde{p}) \, \epsilon \, [0,1]$. In the case of GME, where uniform priors are assumed, *S* may be written as

$$S(\tilde{p}) = \frac{-\tilde{p}' \ln(\tilde{p})}{\sum_i \ln(M_i)}$$

where M_i is the number of support points for parameter i. A value of 0 for *S* implies there is no uncertainty regarding the parameters, hence, it is a degenerate situation. However, a value of 1 implies that the posterior distributions equal the priors, which indicates total uncertainty if the priors are uniform.

Since NE is relative, it may be used for comparing various situations. Consider adding a data point to the model. If $S_{T+1} = S_T$ then there is no additional information contained within that data constraint. However, if $S_{T+1} < S_T$, then the data point will give a more informed set of parameter estimates.

NE can be used for determining the importance of particular variables in regard to the reduction of the uncertainty they bring to the model. Each of the k parameters being estimated has an associated NE defined as

$$S(\tilde{p_k}) = \frac{-\tilde{p}_k' \ln(\tilde{p}_k)}{-\ln(q_k)}$$

or if in the GME case,

$$S(\tilde{p_k}) = \frac{-\tilde{p}_k' \ln(\tilde{p}_k)}{\ln(M)}$$

where p_k is the vector of supports for parameter β_k and m is the corresponding number of support points. Since a value of 1 implies no relative information for that particular sample, Golan, Judge, and Miller (1996) suggest an exclusion criteria of $S(\tilde{p_k}) > 0.99$ as an acceptable means of selecting noninformative variables. See Golan, Judge, and Miller (1996) for some simulation results.

Parameter Covariance For GCE

For the data-constrained cross-entropy problem, the estimate of the asymptotic variance of the signal parameter is given by

$$V\hat{a}r(\hat{\beta}) = \frac{\hat{\sigma_\gamma^2}(\hat{\beta})}{\hat{\psi}^2(\hat{\beta})}(X'X)^{-1}$$

where

$$\hat{\sigma_\gamma^2}(\hat{\beta}) = \frac{1}{N}\sum_{i=1}^{N}\gamma_i^2$$

and γ_i is the Lagrange multiplier associated with the *i*th row of the Vw constraint matrix. Also,

$$\hat{\psi}^2(\hat{\beta}) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{J}v_{ij}^2 w_{ij} - (\sum_{j=1}^{J}v_{ij}w_{ij})^2\right)^{-1}\right]^2$$

Parameter Covariance For GCE-NM

Golan, Judge, and Miller (1996) give the finite approximation to the asymptotic variance matrix of the Normed Moment formulation as

$$V\hat{a}r(\hat{\beta}) = \Sigma_z X'XC^{-1}DC^{-1}X'X\Sigma_z$$

where

$$C = X'X\Sigma_z X'X + \Sigma_v$$

and

$$D = X'\Sigma_e X$$

Recall that in the Normed Moment formulation, V is the support of $\frac{X'e}{T}$, which implies Σ_v is a K-dimensional variance matrix. Σ_z and Σ_v are both diagonal matrices with the form

$$\Sigma_z = \begin{bmatrix} \sum_{l=1}^{L} z_{1l}^2 p_{1l} - (\sum_{l=1}^{L} z_{1l}p_{1l})^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{l=1}^{L} z_{Kl}^2 p_{Kl} - (\sum_{l=1}^{L} z_{Kl}p_{Kl})^2 \end{bmatrix}$$

and

$$\Sigma_v = \begin{bmatrix} \sum_{j=1}^{J} v_{1j}^2 w_{jl} - (\sum_{j=1}^{J} v_{1j} w_{1j})^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{j=1}^{J} v_{Kl}^2 w_{Kl} - (\sum_{j=1}^{J} v_{Kl} w_{Kl})^2 \end{bmatrix}$$

Statistical Tests

Since the GME estimates have been shown to be asymptotically normally distributed, the classical Wald, Score, and Likelihood statistics may be used for testing linear restrictions on the parameters.

Wald Tests

Let $H_0 : L\beta = m$, where L is a set of linearly independent combinations of the elements of β. Then under the null hypothesis, the Wald test statistic,

$$T_W = (L\beta - m)' \left(L(\hat{Var}(\hat{\beta}))L' \right)^{-1} (L\beta - m)$$

has a central χ^2 limiting distribution with degrees of freedom equal to the rank of L.

Pseudo-likelihood Ratio Tests

Using the conditionally maximized entropy function as a pseudo-likelihood, F, Mittelhammer and Cardell (2000) state that

$$\frac{2\hat{\psi}(\hat{\beta})}{\hat{\sigma_\gamma^2}(\hat{\beta})} \left(F(\hat{\beta}) - F(\tilde{\beta}) \right)$$

has the limiting distribution of the Wald statistic when testing the same hypothesis. Note that $F(\hat{\beta})$ and $F(\tilde{\beta})$ are the maximum values of the entropy objective function over the full and restricted parameter spaces, respectively.

Score Tests

Again using the GME function as a pseudo-likelihood, Mittelhammer and Cardell (2000) define the Score statistic as

$$\frac{1}{\hat{\sigma_\gamma^2}(\tilde{\beta})} G(\tilde{\beta})'(X'X)^{-1} G(\tilde{\beta})$$

where G is the gradient of F, which is being evaluated at the optimum point for the restricted parameters. This of course shares the same limiting distribution as the Wald and pseudo-likelihood ratio tests.

Missing Values

If an observation the input data set contains missing value for any of the regressors or dependent values, that observation is dropped.

Output Data Sets

OUT= Data Set

The OUT= data set specified on the ENTROPY statement contains residuals of the dependent variables computed from the parameter estimates. The ID and BY variables are also added to this data set.

OUTEST= Data Set

The OUTEST= data set contains parameter estimates and, if requested, estimates of the covariance of the parameter estimates.

The variables in the data set are as follows:

- BY variables
- _NAME_, a character variable of length 32, blank for observations containing parameter estimates or a parameter name for observations containing covariances
- _TYPE_, a character variable of length 8 identifying the estimation method: GME or GMENM
- the parameters estimated

If the COVOUT option is specified, an additional observation is written for each row of the estimate of the covariance matrix of parameter estimates, with the _NAME_ values containing the parameter names for the rows.

OUTP= Data Set

The OUTP= data set specified on the ENTROPY statement contains the probabilities estimated for each support point, as well as the support points and prior probabilities used in the estimation.

The variables in the data set are as follows:

- BY variables
- _TYPE_, a character variable of length 8 identifying the estimation method: GME or GMENM
- Variable, a character variable of length 32 indicating the name of the regressor. The regressor name and the equation name identify a unique coefficient.
- _OBS_, a numeric variable that is either missing when the probabilities are for coefficients or the observation number when the probabilities are for the residual terms. The _OBS_ and the equation name identify which residual the probability is associated with

- Equation, a character variable of length 32 indicating the name of the dependent variable
- NSupport, a numeric variable indicating the number of support points for each basis. This number is needed for the PDATA= data set.
- Support, a numeric variable that is the support value the probability is associated with
- Prior, a numeric variable that is the prior probability associated with the probability
- Prb, a numeric variable that is the estimated probability

OUTL= Data Set

The OUTL= data set specified on the ENTROPY statement contains the Lagrange multiplier values for the underlying maximum entropy problem.

The variables in the data set are as follows:

- BY variables
- Equation, a character variable of length 32 indicating the name of the dependent variable
- Variable, a character variable of length 32 indicating the name of the regressor. The regressor name and the equation name identify a unique coefficient.
- _OBS_, a numeric variable that is either missing when the probabilities are for coefficients or the observation number when the probabilities are for the residual terms. The _OBS_ and the equation name identify which residual the Lagrange multiplier is associated with
- LagrangeMult, a numeric variable containing the Lagrange multipliers

ODS Table Names

PROC ENTROPY assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 15.1. ODS Tables Produced in PROC ENTROPY

ODS Table Name	**Description**	**Option**
ConvCrit	Convergence criteria for estimation	default
ConvergenceStatus	Convergence status	default
DatasetOptions	Data sets used	default
MinSummary	Number of parameters, estimation kind	default
ObsUsed	Observations read, used, and missing.	default
ParameterEstimates	Parameter Estimates	default
ResidSummary	Summary of the SSE, MSE for the equations	default
TestResults	Test statement table	

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the ENTROPY procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

ODS Graph Names

PROC ENTROPY assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 15.2.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 15.2. ODS Graphics Produced by PROC ENTROPY

ODS Graph Name	Plot Description
ActualByPredicted	Predicted vs actual plot
CooksD	Cook's D plot
StudentResidualPlot	Studentized residual plot

Examples

Example 15.1. Nonnormal Error Estimation

This example illustrates the difference between GME-NM and GME. One of the basic assumptions of OLS estimation is that the errors in the estimation are normally distributed. If this assumption is violated, the estimated parameters are biased. For GME-NM the story is similar. If the first moment of the distribution of the errors and a scale factor cannot be used to describe the distribution, then the parameter estimates from GME-MN will be more biased. GME is much less sensitive to the underlying distribution of the errors than GME-NM.

Consider the following model

$$y = a * x_1 + b * x_2 + \epsilon$$

where ϵ is distributed first normally, then chi-squared with six degrees of freedom, and finally distributed Cauchy.

Here 100 samples of 10 observations each were drawn from each population. The data for the Cauchy errors is generated using the following SAS code:

```
data one;
    call streaminit(156789);
    do by = 1 to 100;
```

```
        do x2 = 1 to 10;
           x1 = 10 * ranuni( 512);
           y = x1 + 2*x2 + rand('cauchy');
           output;
        end;
    end;
run;
```

The code for the other distributions is identical except for the argument to the RAND() function.

The parameters to the model were estimated using data-constrained maximum entropy using the following statements:

```
proc entropy data = one gme outest=parm1;
   model y = x1 x2;
   by by;
run;
```

The estimation using moment-constrained maximum entropy was performed by changing the GME option to GMENM. For comparison the same model was estimated using OLS with the following SAS statements.

```
proc reg data = one outest=parm3;
   model y = x1 x2;
   by by;
run;
```

The 100 estimations of the coefficient on variable `x1` are then summarized using the following SAS statements.

```
proc univariate data=parm1 ;
var x1;
run;
```

The following table summarizes the results from the estimations. The true value for the coefficient on `x1` is 1.0.

Estimation	Normal		Chi-Squared		Cauchy	
Method	Mean	Std Deviation	Mean	Std Deviation	Mean	Std Deviation
GME	0.418	0.117	0.626	.330	0.818	3.36
GME-NM	0.878	0.116	0.948	0.427	3.03	13.62
OLS	0.973	0.142	1.023	0.467	5.54	26.83

Note that for normal or nearly normal data moment-constrained maximum entropy is a good choice. For distributions not well described by a normal distribution, data-constrained maximum entropy is a good choice.

Example 15.2. Unreplicated Factorial Experiments

Overview

Factorial experiments are useful for studying the effects of various factors on a response. For the practitioner constrained to the use of OLS regression, there must be replication to estimate all of the possible main and interaction effects in a factorial experiment. Using OLS regression to analyze unreplicated experimental data will result in no degrees of freedom for error in the ANOVA table, i.e. there are as many parameters as observations. This situation leaves the experimenter unable to compute confidence intervals or perform hypothesis testing on the parameter estimates.

Several options are available when replication is impossible. The higher-order interactions may be assumed to have negligible effects and have their degrees of freedom pooled to create the error degrees of freedom that will be used to perform inference on the lower-order estimates. Or, if a preliminary experiment is being run, a normal probability plot of all effects may give a clue as to which effects are significant and should be focused upon in a later, more complete experiment. The following example illustrates the probability plot methodology and an alternative using PROC ENTROPY. Consider a 2^4 factorial model with no replication. The data are taken from Myers and Montgomery(1995).

```
data rate;
   do a=-1,1; do b=-1,1; do c=-1,1; do d=-1,1;
      input y @@;
      ab=a*b; ac=a*c; ad=a*d; bc=b*c; bd=b*d; cd=c*d;
      abc=a*b*c; abd=a*b*d; acd=a*c*d; bcd=b*c*d;
      abcd=a*b*c*d;
      output;
   end; end; end; end;
   datalines;
   45 71 48 65 68 60 80 65 43 100 45 104 75 86 70 96
   ;
run;
```

Analyze the data using PROC REG, then output the resulting estimates.

```
proc reg data=rate outest=regout;
   model y=a b c d ab ac ad bc bd cd abc abd acd bcd abcd;

proc transpose data=regout out=ploteff name=effect prefix=est;
   var a b c d ab ac ad bc bd cd abc abd acd bcd abcd;
```

Now the normal scores for the estimates must be computed.

```
proc rank data=ploteff normal=blom out=qqplot;
   var est1;
   ranks normalq;
```

To create the probability plot, simply plot the estimates versus their normal scores.

```
proc gplot data=qqplot;
   plot normalq*est1=effect;
```

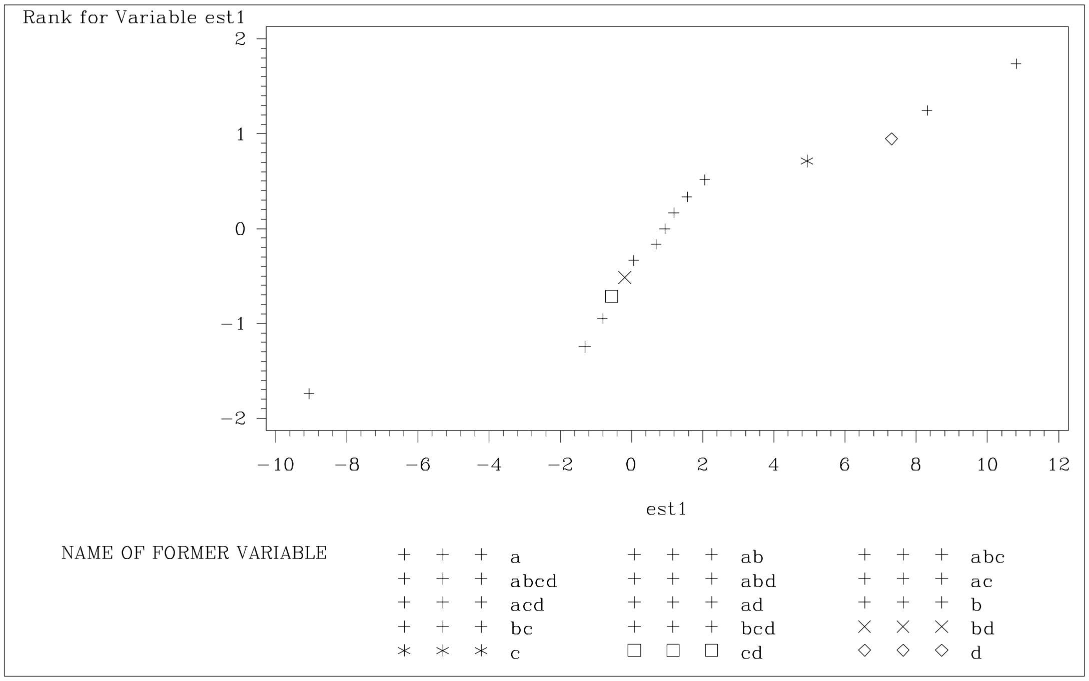


Figure 15.22. Normal Probability Plot of Effects

The plot shown in Figure 15.22 displays evidence that the a, c, d, ac and ad estimates do not fit into the purely random normal model, which suggests that they may have some significant effect on the response variable. To verify this, fit a reduced model that contains only these effects.

```
proc reg data=rate;
   model y=a c d ad ac;
```

The estimates for the reduced model are shown in Output 15.2.1.

Output 15.2.1. Reduced Model OLS Estimates

```
                              The REG Procedure
                                Model: MODEL1
                            Dependent Variable: y

                             Parameter Estimates

                            Parameter       Standard
   Variable      DF          Estimate          Error    t Value    Pr > |t|

   Intercept      1          70.06250        3.40920      20.55      <.0001
   a              1           7.31250        3.40920       2.14      0.0576
   c              1           1.56250        3.40920       0.46      0.6565
   d              1          10.81250        3.40920       3.17      0.0100
   ad             1           8.31250        3.40920       2.44      0.0350
   ac             1          -0.18750        3.40920      -0.05      0.9572
```

These results certainly support the probability plot methodology. However, PROC ENTROPY can directly estimate the full model without having to rely upon the probability plot for clues to which effects may be significant. A simple ENTROPY run using default parameter and error supports is shown.

```
proc entropy data=rate ;
   model y=a b c d ab ac ad bc bd cd abc abd acd bcd abcd;
```

Proc ENTROPY immediately picks out a, c, d, ac and ad as significant. The GME estimates are shown in Output 15.2.2.

Output 15.2.2. Full Model Entropy Results

```
                            The ENTROPY Procedure

                          GME-NM Variable Estimates

                                        Approx                     Approx
       Variable        Estimate        Std Err    t Value       Pr > |t|

       a               5.688835         0.7911       7.19         <.0001
       b               2.988075         0.5464       5.47         <.0001
       c               0.234214         0.1379       1.70         0.1089
       d               9.626745         0.9765       9.86         <.0001
       ab              -0.01384         0.0270      -0.51         0.6156
       ac              -0.00054        0.00325      -0.16         0.8710
       ad                6.8315         0.8628       7.92         <.0001
       bc              0.113876         0.0941       1.21         0.2437
       bd              -7.68139         0.9054      -8.48         <.0001
       cd               0.00002       0.000364       0.05         0.9571
       abc             -0.14882         0.1087      -1.37         0.1898
       abd             -0.03988         0.0516      -0.77         0.4512
       acd             0.467138         0.1960       2.38         0.0299
       bcd             0.059594         0.0654       0.91         0.3755
       abcd             0.02476         0.0387       0.64         0.5317
       Intercept       69.87685         1.1404      61.28         <.0001
```

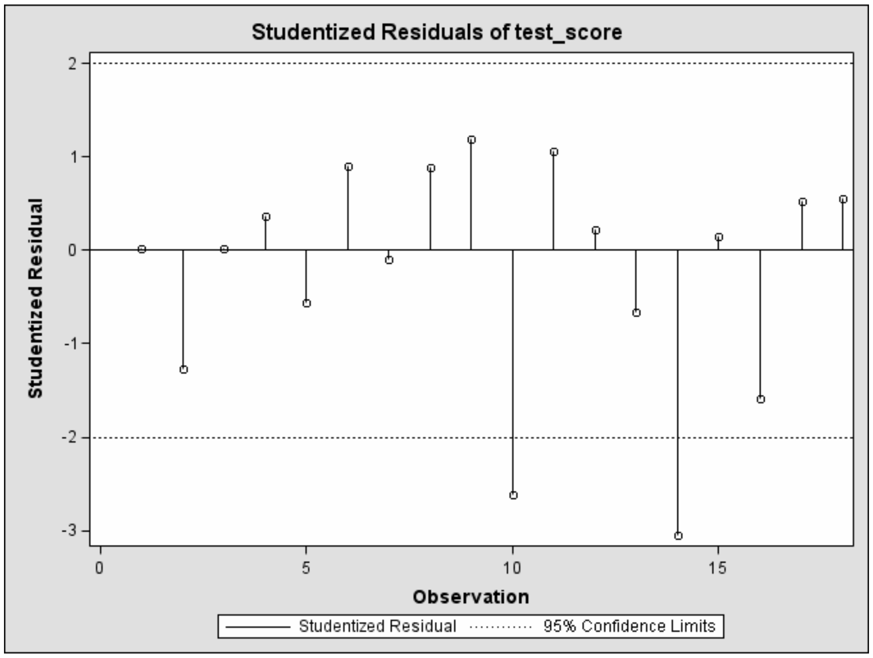
Example 15.3. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics. This is a continuation of the "Simple Regression Analysis" in the section "Getting Started" on page 735. These graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the ENTROPY procedure, see the "ODS Graphics" section on page 774.

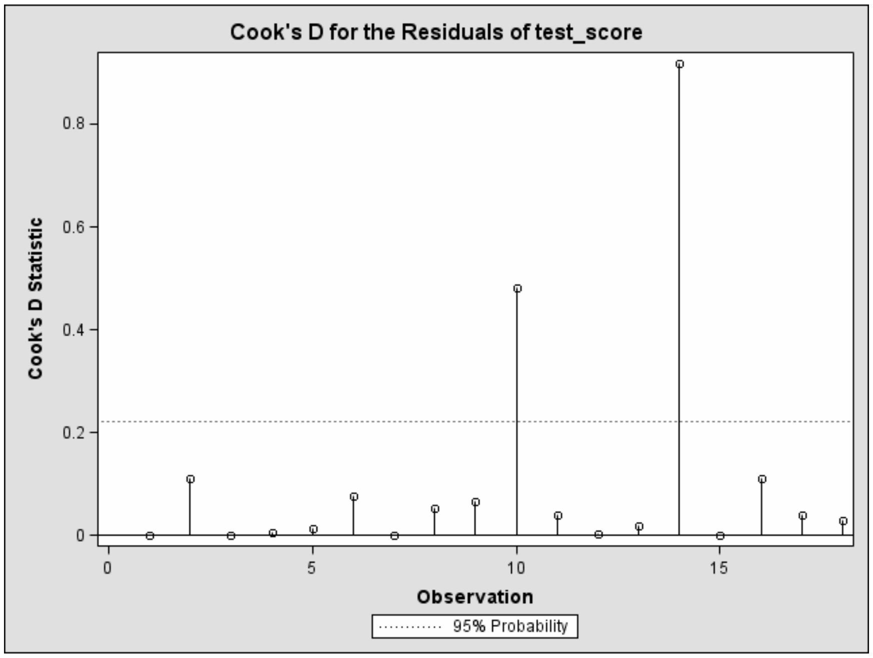
The following statements show how to generate ODS graphics plots with the ENTROPY procedure. The plots are displayed in Output 15.3.1 through Output 15.3.3.

```
ods html;
ods graphics on;

proc entropy data=coleman;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;

ods graphics off;
ods html close;
```

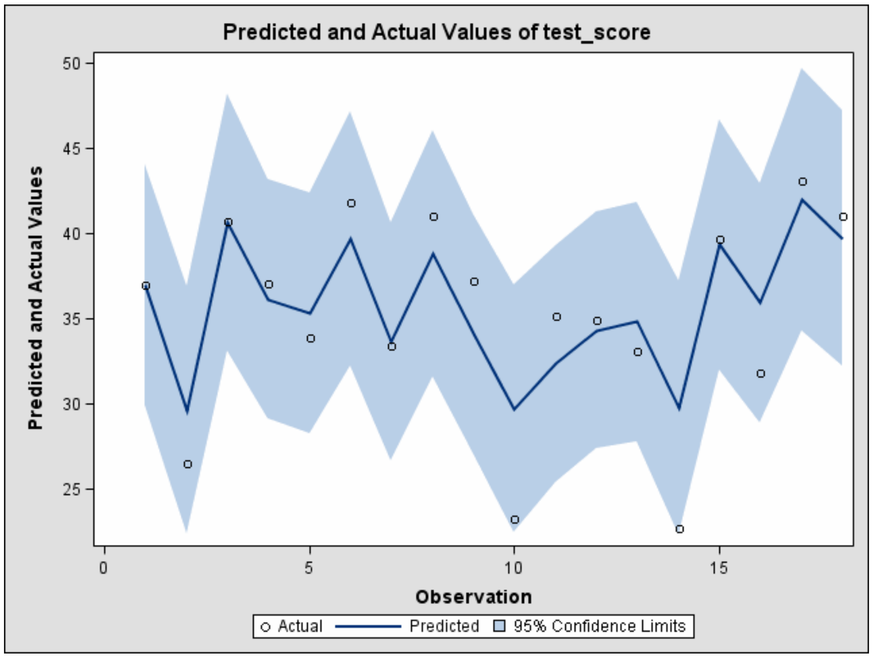
Output 15.3.1. Studentized Residuals Plot (Experimental)



Output 15.3.2. Cook's D Plot (Experimental)



Output 15.3.3. Predicted vs Actual Plot (Experimental)



References

Coleman, J.S., Campbell, E.Q., Hobson, C.J., McPartland, J., Mood, A.M., Weinfeld, F.D., and York, R.L. (1966), *Equality of Educational Opportunity,* Washington, DC: U.S. Government Printing Office.

Deaton, A. and Muellbauer, J. (1980), "An Almost Ideal Demand System," *The American Economic Review*, 70, 312-326.

Eales, J., Prekel, P., and Miller, D., " Quasi-Maximum Likelihood Estimation with Bounded Symmetric Errors", upcoming volume of Advances in Econometrics.

Golan, A., Judge, G., and Miller, D. (1996), *Maximum Entropy Econometrics: Robust Estimation with Limited Data*, Chichester, England: John Wiley & Sons.

Golan, A., Judge, G., and Perloff, J. (1996), "A Generalized Maximum Entropy Approach to Recovering Information From Multinomial Response Data," *Journal of the American Statistical Association*, 91, 841-853.

Good, I.J. (1963), "Maximum Entropy for Hypothesis Formulation, Especially for Multidimensional Contingency Tables," *Annals of Mathematical Statistics*, 34, 911-34.

Kullback, J. (1959), *Information Theory and Statistics*, New York: John Wiley.

Kullback, J., and Leibler, R.A. (1951), "On Information and Sufficiency," *Annals of Mathematical Statistics*, 79-86.

Jaynes, E.T. (1957), "Information of Theory and Statistical Mechanics," *Physics Review*, 106, 620-30.

Jaynes, E.T., (1963), *Information Theory and Statistical Mechanics, in Statistical Physics, K. Ford (ed.),* New York:Benjamin, p. 181.

Kapur, J.N., Kesavan, H.K. (1992), *Entropy Optimization Principles with Applications*, Boston: Academic Press.

LaMotte, L.R. (1994), "A Note on the Role of Independence in t Statistics Constructed from Linear Statistics in Regression Models," *The American Statistician*, 48(3), 238–239.

Mittelhammer, R.C., Cardell S. (2000), "The Data-Constrained GME-estimator of the GLM : Asymptotic Theory and Inference," Working paper of the Department of Statistics, Washington State University, Pullman.

Mittelhammer, R.C., Judge, G.G., and Miller, D.J. (2000), *Econometric Foundations*, Cambridge: Cambridge University Press.

Myers, R. H. and D. Montgomery (1995), "Response Surface Methodology: Process and Product Optimization Using Designed Experiments," New York: John Wiley & Sons.

Shannon, C.E. (1948), "A Mathematical Theory of Communication," *Bell System Technical Journal*, 379-423, 623-659.

Chapter 16
The EXPAND Procedure

Chapter Contents

Chapter 16
The EXPAND Procedure

Overview

The EXPAND procedure converts time series from one sampling interval or frequency to another and interpolates missing values in time series. A wide array of data transformation is also supported. Using PROC EXPAND, you can collapse time series data from higher frequency intervals to lower frequency intervals, or expand data from lower frequency intervals to higher frequency intervals. For example, quarterly estimates can be interpolated from an annual series, or quarterly values can be aggregated to produce an annual series.

Time series frequency conversion is useful when you need to combine series with different sampling intervals into a single data set. For example, if you need as input to a monthly model a series that is only available quarterly, you might use PROC EXPAND to interpolate the needed monthly values.

You can also interpolate missing values in time series, either without changing series frequency or in conjunction with expanding or collapsing the series.

You can convert between any combination of input and output frequencies that can be specified by SAS time interval names. (See Chapter 3, “Date Intervals, Formats, and Functions,” for a complete description of SAS interval names.) When the “from” and “to” intervals are specified, PROC EXPAND automatically accounts for calendar effects such as the differing number of days in each month and leap years.

The EXPAND procedure also handles conversions of frequencies that cannot be defined by standard interval names. Using the FACTOR= option, you can interpolate any number of output observations for each group of a specified number of input observations. For example, if you specify the option FACTOR=(13:2), 13 equally spaced output observations are interpolated from each pair of input observations.

You can also convert aperiodic series, observed at arbitrary points in time, into periodic estimates. For example, a series of randomly timed quality control spot-check results might be interpolated to form estimates of monthly average defect rates.

The EXPAND procedure can also change the observation characteristics of time series. Time series observations can measure beginning-of-period values, end-of-period values, midpoint values, or period averages or totals. PROC EXPAND can convert between these cases. You can construct estimates of interval averages from end-of-period values of a variable, estimate beginning-of-period or midpoint values from interval averages, or compute averages from interval totals, and so forth.

By default, the EXPAND procedure fits cubic spline curves to the nonmissing values of variables to form continuous-time approximations of the input series. Output series are then generated from the spline approximations. Several alternate conversion methods are described in the section “Conversion Methods” on page 802. You can also interpolate estimates of the rate of change of time series by differentiating the interpolating spline curve.

Various transformations can be applied to the input series prior to interpolation and to the interpolated output series. For example, the interpolation process can be modified by transforming the input series, interpolating the transformed series, and applying the inverse of the input transformation to the output series. PROC EXPAND can also be used to apply transformations to time series without interpolation or frequency conversion.

The results of the EXPAND procedure are stored in a SAS data set. No printed output is produced.

Experimental graphics are now available with the EXPAND procedure. For more information, see the "ODS Graphics" section on page 820.

Getting Started

Converting to Higher Frequency Series

To create higher frequency estimates, specify the input and output intervals with the FROM= and TO= options, and list the variables to be converted in a CONVERT statement. For example, suppose variables X, Y, and Z in the data set ANNUAL are annual time series, and you want monthly estimates. You can interpolate monthly estimates by using the following statements:

```
proc expand data=annual out=monthly from=year to=month;
   convert x y z;
run;
```

Note that interpolating values of a time series does not add any real information to the data as the interpolation process is not the same process that generated the other (nonmissing) values in the series. While time series interpolation can sometimes be useful, great care is needed in analyzing time series containing interpolated values.

Aggregating to Lower Frequency Series

PROC EXPAND provides two ways to convert from a higher frequency to a lower frequency. When a curve fitting method is used, converting to a lower frequency is no different than converting to a higher frequency–you just specify the desired output frequency with the TO= option. This provides for interpolation of missing values and allows conversion from nonnested intervals, such as converting from weekly to monthly values.

Alternatively, you can specify simple aggregation or selection without interpolation of missing values. This might be useful, for example, if you wanted to add up monthly values to produce annual totals but wanted the annual output data set to contain values only for complete years.

To perform simple aggregation, use the METHOD=AGGREGATE option in the CONVERT statement. For example, the following statements aggregate monthly values to yearly values:

```
proc expand data=monthly out=annual from=month to=year;
   convert x y z / method=aggregate;
   convert a b c / observed=total method=aggregate;
   id date;
run;
```

Note that the AGGREGATE method can be used only if the input intervals are nested within the output intervals, as when converting from daily to monthly or from monthly to yearly frequency.

Combining Time Series with Different Frequencies

One important use of PROC EXPAND is to combine time series measured at different sampling frequencies. For example, suppose you have data on monthly money stocks (M1), quarterly gross domestic product (GDP), and weekly interest rates (INTEREST), and you want to perform an analysis of a model that uses all these variables. To perform the analysis, you first need to convert the series to a common frequency and combine the variables into one data set.

The following statements illustrate this process for the three data sets QUARTER, MONTHLY, and WEEKLY. The data sets QUARTER and WEEKLY are converted to monthly frequency using two PROC EXPAND steps, and the three data sets are then merged using a DATA step MERGE statement to produce the data set COMBINED.

```
proc expand data=quarter out=temp1 from=qtr to=month;
   id date;
   convert gdp / observed=total;
run;

proc expand data=weekly out=temp2 from=week to=month;
   id date;
   convert interest / observed=average;
run;

data combined;
   merge monthly temp1 temp2;
   by date;
run;
```

See Chapter 2, "Working with Time Series Data," for further discussion of time series periodicity, time series dating, and time series interpolation.

Interpolating Missing Values

To interpolate missing values in time series without converting the observation frequency, leave off the TO= option. For example, the following statements interpolate any missing values in the time series in the data set ANNUAL.

```
proc expand data=annual out=new from=year;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

To interpolate missing values in variables observed at specific points in time, omit both the FROM= and TO= options and use the ID statement to supply time values for the observations. The observations do not need to be periodic or form regular time series, but the data set must be sorted by the ID variable. For example, the following statements interpolate any missing values in the numeric variables in the data set A.

```
proc expand data=a out=b;
   id date;
run;
```

If the observations are equally spaced in time, and all the series are observed as beginning-of-period values, only the input and output data sets need to be specified. For example, the following statements interpolate any missing values in the numeric variables in the data set A, assuming that the observations are at equally spaced points in time.

```
proc expand data=a out=b;
run;
```

Refer to the section “Missing Values” on page 811 for further information.

Requesting Different Interpolation Methods

By default, a cubic spline curve is fit to the input series, and the output is computed from this interpolating curve. Other interpolation methods can be specified with the METHOD= option on the CONVERT statement. The section “Conversion Methods” on page 802 explains the available methods.

For example, the following statements convert annual series to monthly series using linear interpolation instead of cubic spline interpolation.

```
proc expand data=annual out=monthly from=year to=month;
   id date;
   convert x y z / method=join;
run;
```

Using the ID Statement

An ID statement is normally used with PROC EXPAND to specify a SAS date or datetime variable to identify the time of each input observation. An ID variable allows PROC EXPAND to do the following:

- identify the observations in the output data set
- determine the time span between observations and detect gaps in the input series caused by omitted observations
- account for calendar effects such as the number of days in each month and leap years

If you do not specify an ID variable with SAS date or datetime values, PROC EXPAND makes default assumptions that may not be what you want. See the section "ID Statement" for details.

Specifying Observation Characteristics

It is important to distinguish between variables that are measured at points in time and variables that represent totals or averages over an interval. Point-in-time values are often called *stocks* or *levels*. Variables that represent totals or averages over an interval are often called *flows* or *rates*.

For example, the annual series "U.S. Gross Domestic Product" represents the total value of production over the year and also the yearly average rate of production in dollars per year. However, a monthly variable *inventory* may represent the cost of a stock of goods as of the end of the month.

When the data represent periodic totals or averages, the process of interpolation to a higher frequency is sometimes called *distribution*, and the total values of the larger intervals are said to be *distributed* to the smaller intervals. The process of interpolating periodic total or average values to lower frequency estimates is sometimes called *aggregation*.

By default, PROC EXPAND assumes that all time series represent beginning-of-period point-in-time values. If a series does not measure beginning of period point-in-time values, interpolation of the data values using this assumption is not appropriate, and you should specify the correct observation characteristics of the series. The observation characteristics of series are specified with the OBSERVED= option on the CONVERT statement.

For example, suppose that the data set ANNUAL contains variables A, B, and C that measure yearly totals, while the variables X, Y, and Z measure first-of-year values. The following statements estimate the contribution of each month to the annual totals in A, B, and C, and interpolate first-of-month estimates of X, Y, and Z.

```
proc expand data=annual out=monthly from=year to=month;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

The EXPAND procedure supports five different observation characteristics. The OBSERVED= option values for these five observation characteristics are:

BEGINNING	beginning-of-period values
MIDDLE	period midpoint values
END	end-of-period values
TOTAL	period totals
AVERAGE	period averages

The interpolation of each series is adjusted appropriately for its observation characteristics. When OBSERVED=TOTAL or AVERAGE is specified, the interpolating curve is fit to the data values so that the area under the curve within each input interval equals the value of the series. For OBSERVED=MIDDLE or END, the curve is fit through the data points, with the time position of each data value placed at the specified offset from the start of the interval.

See the section “The OBSERVED= Option” on page 794 for details.

Converting Observation Characteristics

The EXPAND procedure can be used to interpolate values for output series with different observation characteristics than the input series. To change observation characteristics, specify two values in the OBSERVED= option. The first value specifies the observation characteristics of the input series; the second value specifies the observation characteristics of the output series.

For example, the following statements convert the period total variable A in the data set ANNUAL to yearly midpoint estimates. This example does not change the series frequency, and the other variables in the data set are copied to the output data set unchanged.

```
proc expand data=annual out=new from=year;
   id date;
   convert a / observed=(total,middle);
run;
```

Creating New Variables

You can use the CONVERT statement to name a new variable to contain the results of the conversion. Using this feature, you can create several different versions of a series in a single PROC EXPAND step. Specify the new name after the input variable name and an equal sign:

```
convert variable=newname ... ;
```

For example, suppose you are converting quarterly data to monthly and you want both first-of-month and midmonth estimates for a beginning-of-period variable X. The following statements perform this task:

```
proc expand data=a out=b from=qtr to=month;
   id date;
   convert x=x_begin  / observed=beginning;
   convert x=x_mid    / observed=(beginning,middle);
run;
```

Transforming Series

The interpolation methods used by PROC EXPAND assume that there are no restrictions on the range of values that series can have. This assumption can sometimes cause problems if the series must be within a certain range.

For example, suppose you are converting monthly sales figures to weekly estimates. Sales estimates should never be less than zero, but since the spline curve ignores this restriction some interpolated values may be negative. One way to deal with this problem is to transform the input series before fitting the interpolating spline and then reverse transform the output series.

You can apply various transformations to the input series using the TRANSFORMIN= option on the CONVERT statement. (The TRANSFORMIN= option can be abbreviated as TRANSFORM= or TIN=.) You can apply transformations to the output series using the TRANSFORMOUT= option. (The TRANSFORMOUT= option can be abbreviated as TOUT=.)

For example, you might use a logarithmic transformation of the input sales series and exponentiate the interpolated output series. The following statements fit a spline curve to the log of SALES and then exponentiate the output series.

```
proc expand data=a out=b from=month to=week;
   id date;
   convert sales / observed=total
                   transformin=(log) transformout=(exp);
run;
```

As another example, suppose you are interpolating missing values in a series of market share estimates. Market shares must be between 0% and 100%, but applying a spline interpolation to the raw series can produce estimates outside of this range.

The following statements use the logistic transformation to transform proportions in the range 0 to 1 to values in the range $-\infty$ to $+\infty$. The TIN= option first divides the market shares by 100 to rescale percent values to proportions and then applies the LOGIT function. The TOUT= option applies the inverse logistic function ILOGIT to the interpolated values to convert back to proportions and then multiplies by 100 to rescale back to percentages.

```
proc expand data=a out=b;
   id date;
   convert mshare / tin=( / 100 logit ) tout=( ilogit * 100 );
run;
```

You can also use the TRANSFORM= (or TRANSFORMOUT=) option as a convenient way to do calculations normally performed with the SAS DATA step. For example, the following statements add the lead of X to the data set A. The METHOD=NONE option is used to suppress interpolation.

```
proc expand data=a method=none;
   id date;
   convert x=xlead / transform=(lead);
run;
```

Any number of operations can be listed in the TRANSFORMIN= and TRANSFORMOUT= options. See Table 16.1 for a list of the operations supported.

Syntax

The EXPAND procedure uses the following statements:

PROC EXPAND *options* **;**
  BY *variables* **;**
  CONVERT *variables / options* **;**
  ID *variable* **;**

Functional Summary

The statements and options controlling the EXPAND procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify conversion options	CONVERT	
specify the ID variable	ID	
Data Set Options		
specify the input data set	PROC EXPAND	DATA=
specify the output data set	PROC EXPAND	OUT=
write interpolating functions to a data set	PROC EXPAND	OUTEST=
extrapolate values before or after input series	PROC EXPAND	EXTRAPOLATE

Description	Statement	Option
Input and Output Frequencies		
specify input frequency	PROC EXPAND	FROM=
specify output frequency	PROC EXPAND	TO=
specify frequency conversion factor	PROC EXPAND	FACTOR=
control the alignment of SAS Date values	PROC EXPAND	ALIGN=
Interpolation Control Options		
specify interpolation method	PROC EXPAND CONVERT	METHOD=
specify observation characteristics	CONVERT	OBSERVED=
specify transformations of the input series	CONVERT	TRANSIN=
specify transformations of the output series	CONVERT	TRANSOUT=

PROC EXPAND Statement

PROC EXPAND *options;*

The following options can be used with the PROC EXPAND statement.

Data Set Options

DATA=*SAS-data-set*
names the input data set. If the DATA= option is omitted, the most recently created SAS data set is used.

OUT=*SAS-data-set*
names the output data set containing the result time series. If OUT= is not specified, the data set is named using the DATA*n* convention. See the section "OUT= Data Set" on page 818 for details.

OUTEST=*SAS-data-set*
names an output data set containing the coefficients of the spline curves fit to the input series. If the OUTEST= option is not specified, the spline coefficients are not output. See the section "OUTEST= Data Set" on page 819 for details.

Options That Define Input and Output Frequencies

FACTOR=*n*
FACTOR=*(n:m)*
FACTOR=*(n,m)*
specifies the number of output observations to be created from the input observations. FACTOR=(*n*:*m*) specifies that *n* output observations are to be produced for each group of *m* input observations. FACTOR=*n* is the same as FACTOR=(*n*:1).

The FACTOR= option cannot be used if the TO= option is used. The default value is FACTOR=(1:1). For more information, see the "Frequency Conversion" section (page 797).

FROM=*interval*
specifies the time interval between observations in the input data set. Examples of FROM= values are YEAR, QTR, MONTH, DAY, and HOUR. See Chapter 3, "Date Intervals, Formats, and Functions," for a complete description and examples of interval specification.

TO=*interval*
specifies the time interval between observations in the output data set. By default, the TO= interval is generated from the combination of the FROM= and the FACTOR= values or is set to be the same as the FROM= value if FACTOR= is not specified. See Chapter 3, "Date Intervals, Formats, and Functions," for a description of interval specifications.

ALIGN=*option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

Options to Control the Interpolation

METHOD=*option*
METHOD=SPLINE(*constraint* **[,** *constraint***])**
specifies the method used to convert the data series. The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the "Conversion Methods" section on page 802 for more information about these methods.

OBSERVED=*value*
OBSERVED=*(from-value, to-value)*
indicates the observation characteristics of the input time series and of the output series. Specifying the OBSERVED= option on the PROC EXPAND statement sets the default OBSERVED= value for subsequent CONVERT statements. See the sections "CONVERT Statement" and "The OBSERVED= Option" later in this chapter for details. The default is OBSERVED=BEGINNING.

EXTRAPOLATE
specifies that missing values at the beginning or end of input series be replaced with values produced by a linear extrapolation of the interpolating curve fit to the input series. See the section "Extrapolation" later in this chapter for details.

By default, PROC EXPAND avoids extrapolating values beyond the first or last input value for a series and only interpolates values within the range of the nonmissing input values. Note that the extrapolated values are often not very accurate, and for the SPLINE method the EXTRAPOLATE option results may be very unreasonable. The EXTRAPOLATE option is not normally used.

BY Statement

BY *variables;*

A BY statement can be used with PROC EXPAND to obtain separate analyses on observations in groups defined by the BY variables. The input data set must be sorted by the BY variables and be sorted by the ID variable within each BY group.

Use a BY statement when you want to interpolate or convert time series within levels of a cross-sectional variable. For example, suppose you have a data set STATE containing annual estimates of average disposable personal income per capita (DPI) by state and you want quarterly estimates by state. These statements convert the DPI series within each state:

```
proc sort data=state;
   by state date;
run;

proc expand data=state out=stateqtr from=year to=qtr;
   convert dpi;
   by state;
   id date;
run;
```

CONVERT Statement

CONVERT *variable=newname ... / options;*

The CONVERT statement lists the variables to be processed. Only numeric variables can be processed.

For each of the variables listed, a new variable name can be specified after an equal sign to name the variable in the output data set that contains the converted values. If a name for the output series is not given, the variable in the output data set has the same name as the input variable. Variable lists may be used only when no name is given for the output series.

For example, variable lists can be specified as follows:

```
convert y1-y25 / observed=(beginning,end);
convert x--a /observed=average;
convert x-numeric-a/observed=average;
```

Any number of CONVERT statements can be used. If no CONVERT statement is used, all the numeric variables in the input data set except those appearing in the BY and ID statements are processed.

The following options can be used with the CONVERT statement.

METHOD=*option*
METHOD=SPLINE(*constraint* **[,** *constraint***])**

specifies the method used to convert the data series. The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the "Conversion Methods" section on page 802 for more information about these methods.

OBSERVED=*value*
OBSERVED=*(from-value, to-value)*

indicates the observation characteristics of the input time series and of the output series. The values supported are TOTAL, AVERAGE, BEGINNING, MIDDLE, and END. In addition, DERIVATIVE can be specified as the *to-value* when the SPLINE method is used. See the section "The OBSERVED= Option" later in this chapter for details.

The default is the value specified for the OBSERVED= option on the PROC EXPAND statement, if any, or else the default value is OBSERVED=BEGINNING.

TRANSFORMIN=*(operation ...)*

specifies a list of transformations to be applied to the input series before the interpolating function is fit. The operations are applied in the order listed. See the section "Transformation Operations" later in this chapter for the operations that can be specified. The TRANSFORMIN= option can be abbreviated as TRANSIN=, TIN=, or TRANSFORM=.

TRANSFORMOUT=*(operation ...)*

specifies a list of transformations to be applied to the output series. The operations are applied in the order listed. See the section "Transformation Operations" later in this chapter for the operations that can be specified. The TRANSFORMOUT= option can be abbreviated as TRANSOUT=, or TOUT=.

ID Statement

ID *variable;*

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values.

The input data must form time series. This means that the observations in the input data set must be sorted by the ID variable (within the BY variables, if any). Moreover, there should be no duplicate observations, and no two observations should have ID values within the same time interval as defined by the FROM= option.

If the ID statement is omitted, SAS date or datetime values are generated to label the input observations. These ID values are generated by assuming that the input data set starts at a SAS date value of 0, that is, 1 January 1960. This default starting date is then incremented for each observation by the FROM= interval (using the INTNX function). If the FROM= option is not specified, the ID values are generated as the observation count minus 1. When the ID statement is not used, an ID variable is added to the output data set named either DATE or DATETIME, depending on the value specified in the TO= option. If neither the TO= option nor the FROM= option is given, the ID variable in the output data set is named TIME.

Details

Frequency Conversion

Frequency conversion is controlled by the FROM=, TO=, and FACTOR= options. The possible combinations of these options are explained in the following:

None Used
If FROM=, TO=, and FACTOR= are not specified, no frequency conversion is done. The data are processed to interpolate any missing values and perform any specified transformations. Each input observation produces one output observation.

FACTOR=(n:m)
FACTOR=(*n*:*m*) specifies that *n* output observations are produced for each group of *m* input observations. The fraction *m*/*n* is reduced first: thus FACTOR=(10:6) is equivalent to FACTOR=(5:3). Note that if *m*/*n*=1, the result is the same as the case given previously under “None Used”.

FROM=interval
The FROM= option used alone establishes the frequency and interval widths of the input observations. Missing values are interpolated, and any specified transformations are performed, but no frequency conversion is done.

TO=interval
When the TO= option is used without the FROM= option, output observations with the TO= frequency are generated over the range of input ID values. The first output observation is for the TO= interval containing the ID value of the first input observation; the last output observation is for the TO= interval containing the ID value of the last input observation. The input observations are not assumed to form regular time series and may represent aperiodic points in time. An ID variable is required to give the date or datetime of the input observations.

FROM=interval TO=interval
When both the FROM= and TO= options are used, the input observations have the frequency given by the FROM= interval, and the output observations have the frequency given by the TO= interval.

FROM=interval FACTOR=(n:m)
When both the FROM= and FACTOR= options are used, a TO= interval is inferred from the combination of the FROM=*interval* and the FACTOR=(*n*:*m*) values specified. For example, FROM=YEAR FACTOR=4 is the same as FROM=YEAR TO=QTR. Also, FROM=YEAR FACTOR=(3:2) is the same as FROM=YEAR used with TO=MONTH8. Once the implied TO= interval is determined, this combination

operates the same as if FROM= and TO= had been specified. If no valid TO= interval can be constructed from the combination of the FROM= and FACTOR= options, an error is produced.

TO=interval FACTOR=(n:m)
The combination of the TO= option and the FACTOR= option is not allowed and produces an error.

ALIGN= option
Controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

Converting to a Lower Frequency

When converting to a lower frequency, the results are either exact or approximate, depending on whether or not the input intervals nest within the output intervals and depending on the need to interpolate missing values within the series. If the TO= interval is nested within the FROM= interval (as when converting monthly to yearly), and if there are no missing input values or partial periods, the results are exact.

When values are missing or the FROM= intervals are not nested within the TO= intervals (as when aggregating weekly to monthly), the results depend on an interpolation. The METHOD=AGGREGATE option always produces exact results, never an interpolation. However, this method cannot be used unless the FROM= interval is nested within the TO= interval.

Identifying Observations

The variable specified in the ID statement is used to identify the observations. Usually, SAS date or datetime values are used for this variable. PROC EXPAND uses the ID variable to do the following:

- identify the time interval of the input values
- validate the input data set observations
- compute the ID values for the observations in the output data set

Identifying the Input Time Intervals

When the FROM= option is specified, observations are understood to refer to the whole time interval and not to a single time point. The ID values are interpreted as identifying the FROM= time interval containing the value. In addition, the widths of these input intervals are used by the OBSERVED= cases TOTAL, AVERAGE, MIDDLE, and END.

For example, if FROM=MONTH is specified, then each observation is for the whole calendar month containing the ID value for the observation, and the width of the time interval covered by the observation is the number of days in that month. Therefore, if FROM=MONTH, the ID value ’31MAR92’D is equivalent to the ID value ’1MAR92’D–both of these ID values identify the same interval, March of 1992.

Widths of Input Time Intervals

When the FROM= option is not specified, the ID variable values are usually interpreted as referring to points in time. However, if an OBSERVED= option is specified that assumes the observations refer to whole intervals and also requires interval widths, then, in the absence of the FROM= specification, interval widths are assumed to be the time span between ID values. For the last observation, the interval width is assumed to be the same as for the next to last observation. (If neither the FROM= option nor the ID statement are specified, interval widths are assumed to be 1.0.) A note is printed in the SAS log warning that this assumption is made.

Validating the Input Data Set Observations

The ID variable is used to verify that successive observations read from the input data set correspond to sequential FROM= intervals. When the FROM= option is not used, PROC EXPAND verifies that the ID values are nonmissing and in ascending order. An error message is produced and the observation is ignored when an invalid ID value is found in the input data set.

ID values for Observations in the Output Data Set

The time unit used for the ID variable in the output data set is controlled by the interval value specified by the TO= option. If you specify a date interval for the TO= value, the ID variable values in the output data set are SAS date values. If you specify a datetime interval for the TO= value, the ID variable values in the output data set are SAS datetime values.

Range of Output Observations

If no frequency conversion is done, the range of output observations is the same as in the input data set.

When frequency conversion is done, the observations in the output data set range from the earliest start of any result series to the latest end of any result series. Observations at the beginning or end of the input range for which all result values are missing are not written to the OUT= data set.

When the EXTRAPOLATE option is not used, the range of the nonmissing output results for each series is as follows. The first result value is for the TO= interval that contains the ID value of the start of the FROM= interval containing the ID value of the first nonmissing input observation for the series. The last result value is for the TO= interval that contains the end of the FROM= interval containing the ID value of the last nonmissing input observation for the series.

When the EXTRAPOLATE option is used, result values for all series are computed for the full time range covered by the input data set.

Extrapolation

The spline functions fit by the EXPAND procedure are very good at approximating continuous curves within the time range of the input data but poor at extrapolating beyond the range of the data. The accuracy of the results produced by PROC EXPAND may be somewhat less at the ends of the output series than at time periods for which there are several input values at both earlier and later times. The curves fit by PROC EXPAND should not be used for forecasting.

PROC EXPAND normally avoids extrapolation of values beyond the time range of the nonmissing input data for a series, unless the EXTRAPOLATE option is used. However, if the start or end of the input series does not correspond to the start or end of an output interval, some output values may depend in part on an extrapolation.

For example, if FROM=YEAR, TO=WEEK, and OBSERVED=BEGINNING, the first observation output for a series is for the week of 1 January of the first nonmissing input year. If 1 January of that year is not a Sunday, the beginning of this week falls before the date of the first input value, and therefore a beginning-of-period output value for this week is extrapolated.

This extrapolation is made only to the extent needed to complete the terminal output intervals that overlap the endpoints of the input series and is limited to no more than the width of one FROM= interval or one TO= interval, whichever is less. This restriction of the extrapolation to complete terminal output intervals is applied to each series separately, and it takes into account the OBSERVED= option for the input and output series.

When the EXTRAPOLATE option is used, the normal restriction on extrapolation is overridden. Output values are computed for the full time range covered by the input data set.

For the SPLINE method, extrapolation is performed by a linear projection of the trend of the cubic spline curve fit to the input data, not by extrapolation of the first and last cubic segments.

The OBSERVED= Option

The values of the CONVERT statement OBSERVED= option are as follows:

BEGINNING	indicates that the data are beginning-of-period values. OBSERVED=BEGINNING is the default.
MIDDLE	indicates that the data are period midpoint values.
ENDING	indicates that the data represent end-of-period values.
TOTAL	indicates that the data values represent period totals for the time interval corresponding to the observation.
AVERAGE	indicates that the data values represent period averages.
DERIVATIVE	requests that the output series be the derivatives of the cubic spline curve fit to the input data by the SPLINE method.

If only one value is specified in the OBSERVED= option, that value applies to both the input and the output series. For example, OBSERVED=TOTAL is the same as OBSERVED=(TOTAL,TOTAL), which indicates both that the input values represent totals over the time intervals corresponding to the input observations and that the converted output values also represent period totals. The value DERIVATIVE can be used only as the second OBSERVED= option value, and it can be used only when METHOD=SPLINE is specified or is the default method.

Since the TOTAL, AVERAGE, MIDDLE, and END cases require that the width of each input interval be known, both the FROM= option and an ID statement are normally required if one of these observation characteristics is specified for any series. However, if the FROM= option is not specified, each input interval is assumed to extend from the ID value for the observation to the ID value of the next observation, and the width of the interval for the last observation is assumed to be the same as the width for the next to last observation.

Scale of OBSERVED=AVERAGE Values

The average values are assumed to be expressed in the time units defined by the FROM= or TO= option. That is, the product of the average value for an interval and the width of the interval is assumed to equal the total value for the interval. For purposes of interpolation, OBSERVED=AVERAGE values are first converted to OBSERVED=TOTAL values using this assumption, and then the interpolated totals are converted back to averages by dividing by the widths of the output intervals. For example, suppose the options FROM=MONTH, TO=HOUR, and OBSERVED=AVERAGE are specified.

Since FROM=MONTH in this example, each input value is assumed to represent an average rate per day such that the product of the value and the number of days in the month is equal to the total for the month. The input values are assumed to represent a per-day rate because FROM=MONTH implies SAS date ID values that measure time in days, and therefore the widths of MONTH intervals are measured in days. If FROM=DTMONTH is used instead, the values are assumed to represent a per-second rate, because the widths of DTMONTH intervals are measured in seconds.

Since TO=HOUR in this example, the output values are scaled as an average rate per second such that the product of each output value and the number of seconds in an hour (3600) is equal to the interpolated hourly total. A per-second rate is used because TO=HOUR implies SAS datetime ID values that measure time in seconds, and therefore the widths of HOUR intervals are measured in seconds.

Note that the scale assumed for OBSERVED=AVERAGE data is important only when converting between AVERAGE and another OBSERVED= option, or when converting between SAS date and SAS datetime ID values. When both the input and the output series are AVERAGE values, and the units for the ID values are not changed, the scale assumed does not matter.

For example, suppose you are converting a series gross domestic product (GDP) from quarterly to monthly. The GDP values are quarterly averages measured at annual rates. If you want the interpolated monthly values to also be measured at annual rates, then the option OBSERVED=AVERAGE works fine. Since there is no change of scale involved in this problem, it makes no difference that PROC EXPAND assumes daily rates instead of annual rates.

However, suppose you want to convert GDP from quarterly to monthly and also convert from annual rates to monthly rates, so that the result is total gross domestic product for the month. Using the option OBSERVED=(AVERAGE,TOTAL) would fail, because PROC EXPAND assumes the average is scaled to daily, not annual, rates.

One solution is to rescale to quarterly totals and treat the data as totals. You could use the options TRANSFORMIN=(/ 4) OBSERVED=TOTAL. Alternatively, you could treat the data as averages but first convert to daily rates. In this case you would use the options TRANSFORMIN=(/ 365.25) OBSERVED=AVERAGE.

Results of the OBSERVED=DERIVATIVE Option

If the first value of the OBSERVED= option is BEGINNING, TOTAL, or AVERAGE, the result is the derivative of the spline curve evaluated at first-of-period ID values for the output observation. For OBSERVED=(MIDDLE,DERIVATIVE), the derivative of the function is evaluated at output interval midpoints. For OBSERVED=(END,DERIVATIVE), the derivative is evaluated at end-of-period ID values.

Conversion Methods

The SPLINE Method

The SPLINE method fits a cubic spline curve to the input values. A cubic spline is a segmented function consisting of third-degree (cubic) polynomial functions joined together so that the whole curve and its first and second derivatives are continuous.

For point-in-time input data, the spline curve is constrained to pass through the given data points. For interval total or average data, the definite integrals of the spline over the input intervals are constrained to equal the given interval totals.

For boundary constraints, the *not-a-knot* condition is used by default. This means that the first two spline pieces are constrained to be part of the same cubic curve, as are the last two pieces. Thus the spline used by PROC EXPAND by default is not the same as the commonly used natural spline, which uses zero second-derivative endpoint constraints. While DeBoor (1981) recommends the *not-a-knot* constraint for cubic spline interpolation, using this constraint can sometimes produce anomalous results at the ends of the interpolated series. PROC EXPAND provides options to specify other endpoint constraints for spline curves.

To specify endpoint constraints, use the following form of the METHOD= option.

METHOD=SPLINE(*constraint* **[,** *constraint***])**

The first constraint specification applies to the lower endpoint, and the second constraint specification applies to the upper endpoint. If only one constraint is specified, it applies to both the lower and upper endpoints.

The *constraint* specifications can have the following values:

NOTANOT
specifies the not-a-knot constraint. This is the default.

NATURAL
specifies the *natural spline* constraint. The second derivative of the spline curve is constrained to be zero at the endpoint.

SLOPE= value
specifies the first derivative of the spline curve at the endpoint.

CURVATURE= value
specifies the second derivative of the spline curve at the endpoint. Specifying CURVATURE=0 is equivalent to specifying the NATURAL option.

For example, to specify natural spline interpolation, use the following option in the CONVERT or PROC EXPAND statement:

```
method=spline(natural)
```

For OBSERVED=BEGINNING, MIDDLE, and END series, the spline knots are placed at the beginning, middle, and end of each input interval, respectively. For total or averaged series, the spline knots are set at the start of the first interval, at the end of the last interval, and at the interval midpoints, except that there are no knots for the first two and last two midpoints.

Once the cubic spline curve is fit to the data, the spline is extended by adding linear segments at the beginning and end. These linear segments are used for extrapolating values beyond the range of the input data.

For point-in-time output series, the spline function is evaluated at the appropriate points. For interval total or average output series, the spline function is integrated over the output intervals.

The JOIN Method

The JOIN method fits a continuous curve to the data by connecting successive straight line segments. (This produces a linear spline.) For point-in-time data, the JOIN method connects successive nonmissing input values with straight lines. For interval total or average data, interval midpoints are used as the break points, and ordinates are chosen so that the integrals of the piecewise linear curve agree with the input totals.

For point-in-time output series, the JOIN function is evaluated at the appropriate points. For interval total or average output series, the JOIN function is integrated over the output intervals.

The STEP Method

The STEP method fits a discontinuous piecewise constant curve. For point-in-time input data, the resulting step function is equal to the most recent input value. For interval total or average data, the step function is equal to the average value for the interval.

For point-in-time output series, the step function is evaluated at the appropriate points. For interval total or average output series, the step function is integrated over the output intervals.

The AGGREGATE Method

The AGGREGATE method performs simple aggregation of time series without interpolation of missing values.

If the input data are totals or averages, the results are the sums or averages, respectively, of the input values for observations corresponding to the output observations. That is, if either TOTAL or AVERAGE is specified for the OBSERVED= option, the METHOD=AGGREGATE result is the sum or mean of the input values corresponding to the output observation. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR. For OBSERVED=TOTAL series, the result for each output year is the sum of the input values over the months of that year. If any input value is missing, the corresponding sum or mean is also a missing value.

If the input data are point-in-time values, the result value of each output observation equals the input value for a selected input observation determined by the OBSERVED= attribute. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR. For OBSERVED=BEGINNING series, January observations are selected as the annual values. For OBSERVED=MIDDLE series, July observations are selected as the annual values. For OBSERVED=END series, December observations are selected as the annual values. If the selected value is missing, the output annual value is missing.

The AGGREGATE method can be used only when the FROM= intervals are nested within the TO= intervals. For example, you can use METHOD=AGGREGATE when FROM=MONTH and TO=QTR because months are nested within quarters. You cannot use METHOD=AGGREGATE when FROM=WEEK and TO=QTR because weeks are not nested within quarters.

In addition, the AGGREGATE method cannot convert between point-in-time data and interval total or average data. Conversions between TOTAL and AVERAGE data are allowed, but conversions between BEGINNING, MIDDLE, and END are not.

Missing input values produce missing result values for METHOD=AGGREGATE. However, gaps in the sequence of input observations are not allowed. For example, if FROM=MONTH, you may have a missing value for a variable in an observation for a given February. But if an observation for January is followed by an observation for March, there is a gap in the data, and METHOD=AGGREGATE cannot be used.

When the AGGREGATE method is used, there is no interpolating curve, and therefore the EXTRAPOLATE option is not allowed.

METHOD=NONE

The option METHOD=NONE specifies that no interpolation be performed. This option is normally used in conjunction with the TRANSFORMIN= or TRANSFORMOUT= option.

When METHOD=NONE is specified, there is no difference between the TRANSFORMIN= and TRANSFORMOUT= options; if both are specified, the TRANSFORMIN= operations are performed first, followed by the

TRANSFORMOUT= operations. TRANSFORM= can be used as an abbreviation for TRANSFORMIN=. METHOD=NONE cannot be used when frequency conversion is specified.

Transformation Operations

The operations that can be used in the TRANSFORMIN= and TRANSFORMOUT= options are shown in Table 16.1. Operations are applied to each value of the series. Each value of the series is replaced by the result of the operation.

In Table 16.1, x_t or *x* represents the value of the series at a particular time period *t* before the transformation is applied, y_t represents the value of the result series, and N represents the total number of observations.

The notation [*n*] indicates that the argument *n* is optional; the default is 1. The notation *window* is used as the argument for the moving statistics operators, and it indicates that you can specify either an integer number of periods *n* or a list of *n* weights in parentheses. The notation *sequence* is used as the argument for the sequence operators, and it indicates that you must specify a sequence of numbers. The notation *s* indicates the length of seasonality, and it is a required argument.

Table 16.1. Transformation Operations

Syntax	**Result**
+ *number*	adds the specified *number*: $x + number$
- *number*	subtracts the specified *number*: $x - number$
* *number*	multiplies by the specified *number*: $x * number$
& *number*	divides by the specified *number*: $x \& number$
ADJUST	indicates that the following moving window summation or product operator should be adjusted for window width
ABS	absolute value: $\lvert x \rvert$
CD_I *s*	classical decomposition irregular component
CD_S *s*	classical decomposition seasonal component
CD_SA *s*	classical decomposition seasonally adjusted series
CD_TC *s*	classical decomposition trend-cycle component
CDA_I *s*	classical decomposition (additive) irregular component
CDA_S *s*	classical decomposition (additive) seasonal component
CDA_SA *s*	classical decomposition (additive) seasonally adjusted series
CEIL	smallest integer greater than or equal to *x*: $\mathrm{ceil(x)}$
CMOVAVE *window*	centered moving average
CMOVCSS *window*	centered moving corrected sum of squares
CMOVGMEAN *window*	centered moving geometric mean
CMOVMAX *n*	centered moving maximum
CMOVMED *n*	centered moving median
CMOVMIN *n*	centered moving minimum
CMOVPROD *window*	centered moving product
CMOVRANGE *n*	centered moving range
CMOVRANK n	centered moving rank

Table 16.1. (continued)

Syntax	**Result**
CMOVSTD *window*	centered moving standard deviation
CMOVSUM *n*	centered moving sum
CMOVTVALUE *window*	centered moving *t*-value
CMOVUSS *window*	centered moving uncorrected sum of squares
CMOVVAR *window*	centered moving variance
CUAVE [*n*]	cumulative average
CUCSS [*n*]	cumulative corrected sum of squares
CUGMEAN	cumulative geometric mean
CUMAX [*n*]	cumulative maximum
CUMED [*n*]	cumulative median
CUMIN [*n*]	cumulative minimum
CUPROD	cumulative product
CURANK	cumulative rank
CURANGE [*n*]	cumulative range
CUSTD [*n*]	cumulative standard deviation
CUSUM [*n*]	cumulative sum
CUTVALUE	cumulative *t*-value
CUUSS [*n*]	cumulative uncorrected sum of squares
CUVAR [*n*]	cumulative variance
DIF [*n*]	lag *n* difference: $x_t - x_{t-n}$
EWMA *number*	exponentially weighted moving average of x with smoothing weight *number*, where $0 < number < 1$: $y_t = number\ \ x_t + (1 - number)y_{t-1}$. This operation is also called simple exponential smoothing.
EXP	exponential function: $\exp(x)$
FDIF *d*	fractional difference with difference order d where $0 < d < 0.5$
FLOOR	largest integer less than or equal to *x*: $\mathrm{floor(x)}$
FSUM d	fractional summation with summation order d where $0 < d < 0.5$
HP_T *lambda*	Hodrick-Prescott Filter trend component where *lambda* is the non-negative filter parameter
HP_C *lambda*	Hodrick-Prescott Filter cycle component where *lambda* is the non-negative filter parameter
ILOGIT	inverse logistic function: $\frac{\exp(x)}{1+\exp(x)}$
LAG [*n*]	value of the series *n* periods earlier: x_{t-n}
LEAD [*n*]	value of the series *n* periods later: x_{t+n}
LOG	natural logarithm: $\log(x)$
LOGIT	logistic function: $\log(\frac{x}{1-x})$
MAX *number*	maximum of *x* and *number*: $\max(x, number)$
MIN *number*	minimum of *x* and *number*: $\min(x, number)$
> *number*	missing value if $x <= number$, else *x*
>= *number*	missing value if $x < number$, else *x*
= *number*	missing value if $x \neq number$, else *x*

Table 16.1. (continued)

Syntax	**Result**
∧= *number*	missing value if $x = number$, else *x*
< *number*	missing value if $x >= number$, else *x*
<= *number*	missing value if $x > number$, else *x*
MOVAVE *n*	moving average of *n* neighboring values: $\frac{1}{n}\sum_{j=0}^{n-1} x_{t-j}$
MOVAVE($w_1 \ldots w_n$)	weighted moving average of neighboring values: $(\sum_{j=1}^{n} w_j x_{t-j+1})/(\sum_{j=1}^{n} w_j)$
MOVAVE *window*	backward moving average
MOVCSS *window*	backward moving corrected sum of squares
MOVMAX *n*	backward moving maximum
MOVGMEAN *window*	backward moving geometric mean
MOVMED *n*	backward moving median
MOVMIN *n*	backward moving minimum
MOVPROD *window*	backward moving product
MOVRANK n	backward moving rank
MOVRANGE *n*	backward moving range
MOVSTD *window*	backward moving standard deviation
MOVSUM *n*	backward moving sum
MOVTVALUE *window*	backward moving *t*-value
MOVUSS *window*	backward moving uncorrected sum of squares
MOVVAR *window*	backward moving variance
MISSONLY <MEAN>	indicates that the following moving time window statistic operator should replace only missing values with the moving statistic and should leave nonmissing values unchanged. If the option MEAN is specified, then missing values are replaced by the overall mean of the series.
NEG	changes the sign: $-x$
NOMISS	indicates that the following moving time window statistic operator should not allow missing values
PCTDIF n	percent difference of the current value and lag n
PCTSUM n	percent summation of the current value and cumulative sum n-lag periods
RATIO n	ratio of current value to lag n
RECIPROCAL	reciprocal: $1/x$
REVERSE	reverse the series: x_{N-t}
SCALE n_1 n_2	scale series between n_1 and n_2
SEQADD *sequence*	add sequence values to series
SEQDIV *sequence*	divide series by sequence values
SEQMINUS *sequence*	subtract sequence values to series
SEQMULT *sequence*	multiply series by sequence values
SET n_1 n_2	set all values of n_1 to n_2
SETEMBEDDED n_1 n_2	set embedded values of n_1 to n_2
SETLEFT n_1 n_2	set beginning values of n_1 to n_2
SETMISS *number*	replaces missing values in the series with the number specified

Table 16.1. (continued)

Syntax	**Result**
SETRIGHT n_1 n_2	set ending values of n_1 to n_2
SIGN	-1, 0, or 1 as *x* is < 0, equals 0, or > 0 respectively
SQRT	square root: $\sqrt{x}$
SQUARE	square: x^2
SUM	cumulative sum: $\sum_{j=1}^{t} x_j$
SUM *n*	cumulative sum of *n*-period lags: $x_t + x_{t-n} + x_{t-2n} + \ldots$
TRIM *n*	sets x_t to missing a value if $t \leq n$ or $t \geq N - n + 1$
TRIMLEFT *n*	sets x_t to missing a value if $t \leq n$
TRIMRIGHT *n*	sets x_t to missing a value if $t \geq N - n + 1$

Moving Time Window Operators

Some operators compute statistics for a set of values within a moving time window; these are called *moving time window operators*. There are backward and centered versions of these operators.

The centered moving time window operators are CMOVAVE, CMOVCSS, CMOVGMEAN, CMOVMAX, CMOVMED, CMOVMIN, CMOVPROD, CMOVRANGE, CMOVRANK, CMOVSTD, CMOVTVALUE, CMOVSUM, CMOVUSS, and CMOVVAR. These operators compute statistics of the n values x_i for observations $t-(n-1)/2 \ \leq \ i \ \leq \ t+(n-1)/2$.

The backward moving time window operators are MOVAVE, MOVCSS, MOVGMEAN, MOVMAX, MOVMED, MOVMIN, MOVPROD, MOVRANGE, MOVRANK, MOVSTD, MOVTVALUE, MOVSUM, MOVUSS, and MOVVAR. These operators compute statistics of the n values $x_t, x_{t-1}, \ldots, x_{t-n+1}$.

All the moving time window operators accept an argument n specifying the number of periods to include in the time window. For example, the following statement computes a five-period backward moving average of *X*.

```
convert x=y / transformout=( movave 5 );
```

In this example, the resulting transformation is
$y_t = (x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4})/5$.

The following statement computes a five-period centered moving average of *X*.

```
convert x=y / transformout=( cmovave 5 );
```

In this example, the resulting transformation is
$y_t = (x_{t-2} + x_{t-1} + x_t + x_{t+1} + x_{t+2})/5$.

If the window with a centered moving time window operator is not an odd number, one more lagged value than lead value is included in the time window. For example,

the result of the CMOVAVE 4 operator is
$y_t = (x_{t-1} + x_t + x_{t+1} + x_{t+2})/4$.

You can compute a forward moving time window operation by combining a backward moving time window operator with the REVERSE operator. For example, the following statement computes a five-period forward moving average of *X*.

```
convert x=y / transformout=( reverse movave 5 reverse );
```

In this example, the resulting transformation is
$y_t = (x_t + x_{t+1} + x_{t+2} + x_{t+3} + x_{t+4})/5$.

Some of the moving time window operators enable you to specify a list of weight values to compute weighted statistics. These are CMOVAVE, CMOVCSS, CMOVGMEAN, CMOVPROD, CMOVSTD, CMOVTVALUE, CMOVUSS, CMOVVAR, MOVAVE, MOVCSS, MOVGMEAN, MOVPROD, MOVSTD, MOVTVALUE, MOVUSS, and MOVVAR.

To specify a weighted moving time window operator, enter the weight values in parentheses after the operator name. The window width n is equal to the number of weights that you specify; do not specify n.

For example, the following statement computes a weighted five-period centered moving average of *X*.

```
convert x=y / transformout=( cmovave( .1 .2 .4 .2 .1 ) );
```

In this example, the resulting transformation is
$y_t = .1x_{t-2} + .2x_{t-1} + .4x_t + .2x_{t+1} + .1x_{t+2}$.

The weight values must be greater than zero. If the weights do not sum to 1, the weights specified are divided by their sum to produce the weights used to compute the statistic.

At the beginning of the series, and at the end of the series for the centered operators, a complete time window is not available. The computation of the moving time window operators is adjusted for these boundary conditions as follows.

For backward moving window operators, the width of the time window is shortened at the beginning of the series. For example, the results of the MOVSUM 3 operator are

$$\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_1 + x_2 \\
y_3 &= x_1 + x_2 + x_3 \\
y_4 &= x_2 + x_3 + x_4 \\
y_5 &= x_3 + x_4 + x_5 \\
&\dots
\end{aligned}$$

For centered moving window operators, the width of the time window is shortened at the beginning and the end of the series due to unavailable observations. For example, the results of the CMOVSUM 5 operator are

$$\begin{aligned}
y_1 &= x_1 + x_2 + x_3 \\
y_2 &= x_1 + x_2 + x_3 + x_4 \\
y_3 &= x_1 + x_2 + x_3 + x_4 + x_5 \\
y_4 &= x_2 + x_3 + x_4 + x_5 + x_6 \\
&\ldots \\
y_{N-2} &= x_{N-4} + x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_{N-1} &= x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_N &= x_{N-2} + x_{N-1} + x_N
\end{aligned}$$

For weighted moving time window operators, the weights for the unavailable or unused observations are ignored and the remaining weights renormalized to sum to 1.

Cumulative Statistics Operators

Some operators compute cumulative statistics for a set of current and previous values of the series. The cumulative statistics operators are CUAVE, CUCSS, CUMAX, CUMED, CUMIN, CURANGE, CUSTD, CUSUM, CUUSS, and CUVAR. These operators compute statistics of the values $x_t, x_{t-n}, x_{t-2n}, \ldots, x_{t-in}$ for $t - in > 0$.

By default, the cumulative statistics operators compute the statistics from all previous values of the series, so that y_t is based on the set of values $x_1, x_2, \ldots, x_t$. For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for $i \le t$.

```
convert x=y / transformout=( cusum );
```

You can also specify a lag increment argument n for the cumulative statistics operators. In this case, the statistic is computed from the current and every n^{th} previous value. For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for odd i when t is odd and for even i when t is even.

```
convert x=y / transformout=( cusum 2 );
```

The results of this example are

$$\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_2 \\
y_3 &= x_1 + x_3 \\
y_4 &= x_2 + x_4 \\
y_5 &= x_1 + x_3 + x_5
\end{aligned}$$

$$
\begin{aligned}
y_6 &= x_2 + x_4 + x_6 \\
&\dots
\end{aligned}
$$

Missing Values

You can truncate the length of the result series by using the TRIM, TRIMLEFT, and TRIMRIGHT operators to set values to missing at the beginning or end of the series.

You can use these functions to trim the results of moving time window operators so that the result series contains only values computed from a full width time window. For example, the following statements compute a centered five-period moving average of *X*, and they set to missing values at the ends of the series that are averages of fewer than five values.

```
convert x=y / transformout=( cmovave 5 trim 2 );
```

Normally, the moving time window and cumulative statistics operators ignore missing values and compute their results for the nonmissing values. When preceded by the NOMISS operator, these functions produce a missing result if any value within the time window is missing.

The NOMISS operator does not perform any calculations, but serves to modify the operation of the moving time window operator that follows it. The NOMISS operator has no effect unless it is followed by a moving time window operator.

For example, the following statement computes a five-period moving average of the variable *X* but produces a missing value when any of the five values are missing.

```
convert x=y / transformout=( nomiss movave 5 );
```

The following statement computes the cumulative sum of the variable *X* but produces a missing value for all periods after the first missing X value.

```
convert x=y / transformout=( nomiss cusum );
```

Similar to the NOMISS operator, the MISSONLY operator does not perform any calculations (unless followed by the MEAN option), but it serves to modify the operation of the moving time window operator that follows it. When preceded by the MISSONLY operator, these moving time window operators replace any missing values with the moving statistic and leave nonmissing values unchanged.

For example, the following statement replaces any missing values of the variable *X* with an exponentially weighted moving average of the past values of *X* and leaves nonmissing values unchanged. The missing values are then interpolated using an exponentially weighted moving average or simple exponential smoothing.

```
convert x=y / transformout=( missonly ewma 0.3 );
```

For example, the following statement replaces any missing values of the variable *X* with the overall mean of *X*.

```
convert x=y / transformout=( missonly mean );
```

You can use the SETMISS operator to replace missing values with a specified number. For example, the following statement replaces any missing values of the variable *X* with the number 8.77.

```
convert x=y / transformout=( setmiss 8.77 );
```

Classical Decomposition Operators

If y_t is a seasonal time series with s observations per season, *classical decomposition* methods "break down" a time series into four components: trend, cycle, seasonal, and irregular components. The trend and cycle components are often combined to form the trend-cycle component. There are two forms of decomposition: multiplicative and additive.

$$\begin{aligned} y_t &= TC_t S_t I_t \\ y_t &= TC_t + S_t + I_t \end{aligned}$$

where

TC_t	is the trend-cycle component
S_t	is the seasonal component or seasonal factors that are periodic with period s and with mean one (multiplicative) or zero (additive)
I_t	is the irregular or random component that is assumed to have mean one (multiplicative) or zero (additive)

The CD_TC operator computes the trend-cycle component for both the multiplicative and additive models. When s is odd, this operator computes an s-period centered moving average as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} y_{t+k}/s$$

In the case $s = 5$, the CD_TC *s* operator is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=( cmovave 5 trim 2 );
```

When s is even, the CD_TC *s* operator computes the average of two adjacent s-period centered moving averages as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor - 1} (y_{t+k} + y_{t+1+k})/2s$$

In the case $s = 12$, the CD_TC *s* operator is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=(cmovave 12 movave 2 trim 6);
```

The CD_S and CDA_S operators compute the seasonal components for the multiplicative and additive models, respectively. First, the trend-cycle component is computed as shown previously. Second, the seasonal-irregular component is computed by $SI_t = y_t/TC_t$ for the multiplicative model and by $SI_t = y_t - TC_t$ for the additive model. The seasonal component is obtained by averaging the seasonal-irregular component for each season.

$$S_{k+js} = \sum_{t=k \bmod s} \frac{SI_t}{n/s}$$

where $0 \le j \le n/s$ and $1 \le k \le s$. The seasonal components are normalized to sum to one (multiplicative) or zero (additive).

The CD_I and CDA_I operators compute the irregular component for the multiplicative and additive models respectively. First, the seasonal component is computed as shown previously. Next, the irregular component is determined from the seasonal-irregular and seasonal components as appropriate.

$$\begin{aligned} I_t &= SI_t/S_t \\ I_t &= SI_t - S_t \end{aligned}$$

The CD_SA and CDA_SA operators compute the seasonally adjusted time series for the multiplicative and additive models, respectively. After decomposition, the original time series can be seasonally adjusted as appropriate.

$$\begin{aligned} \tilde{y}_t &= y_t/S_t = TC_t I_t \\ \tilde{y}_t &= y_t - S_t = TC_t + I_t \end{aligned}$$

The following statements compute all the multiplicative classical decomposition components for the variable *X* for s=12.

```
convert x=tc / transformout=( cd_tc 12 );
convert x=s  / transformout=( cd_s  12 );
convert x=i  / transformout=( cd_i  12 );
convert x=sa / transformout=( cd_sa 12 );
```

The following statements compute all the additive classical decomposition components for the variable *X* for s=4.

```
convert x=tc / transformout=( cd_tc  4 );
convert x=s  / transformout=( cda_s  4 );
convert x=i  / transformout=( cda_i  4 );
convert x=sa / transformout=( cda_sa 4 );
```

Fractional Operators

For fractional operators, the parameter, *d*, represents the order of fractional differencing. Fractional summation is the inverse operation of fractional differencing.

Examples of Usage

Suppose that *Y* is a fractionally integrated time series variable of order d=0.25. Fractionally differencing *Y* forms a time series variable *X* which is not integrated.

```
convert y=x / transformout=(fdif 0.25);
```

Suppose that *Z* is a nonintegrated time series variable. Fractionally summing *Z* forms a time series W which is fractionally integrated of order $d = 0.25$.

```
convert z=w / transformout=(fsum 0.25);
```

Moving Rank Operators

For the rank operators, the ranks are computed based on the current value with respect to the cumulative, centered, or moving window values. If the current value is missing, the transformed current value is set to missing. If the NOMISS option was previously specified and if any missing values are present in the moving window, the transformed current value is set to missing. Otherwise, redundant values from the moving window are removed and the rank of the current value is computed among the unique values of the moving window.

Examples of Usage

The trades of a particular security are recorded for each weekday in a variable named PRICE. Given the historical daily trades, what is the ranking of the price of this security for each trading day considering its entire past history?

```
convert price=history / transformout=( curank );
```

What is the ranking of the price of this security for each trading day considering the previous week's history?

```
convert price=lastweek / transformout=( movrank 5 );
```

What is the ranking of the price of this security for each trading day considering the previous two week's history?

```
convert price=twoweek / transformout=( movrank 10 );
```

Moving Product Operators

For the product operators, the current transformed value is computed based on the (weighted) product of the cumulative, centered, or moving window values. If missing values are present in the moving window and the NOMISS operator is previously specified, the current transformed value is set to missing. Otherwise, the current transformed value is set to the (weighted) product of the nonmissing values of the moving window. If the geometric mean is desired, the exponents of each product are normalized to one.

Examples of Usage

The interest rates for a savings account are recorded for each month in the data set variable RATES. What is the cumulative interest rate for each month considering the entire account past history?

```
convert rates=history / transformout=( + 1 cuprod - 1);
```

What is the interest rate for each quarter considering the previous quarter history?

```
convert rates=lastqtr / transformout=( + 1 movprod 3 - 1);
```

Sequence Operators

For the sequence operators, the sequence values are used to compute the transformed values from the original values in a sequential fashion. You can add to or subtract from the original series or you can multiply or divide by the sequence values. The first sequence value is applied to the first observation of the series, the second sequence value is applied to the second observation of the series, and so on until the end of the sequence is reached. At this point, the first sequence value is applied to the next observation of the series and the second sequence value on the next observation and so on.

Let $v_1, \ldots, v_m$ be the sequence values and let x_t, $t = 1, \ldots N$, be the original time series. The transformed series, y_t, is computed as follows:

$$\begin{aligned} y_1 &= x_1 \; op \; v_1 \\ y_2 &= x_2 \; op \; v_2 \\ &\ldots \end{aligned}$$

$$\begin{aligned}
y_m &= x_m \; op \; v_m \\
y_{m+1} &= x_{m+1} \; op \; v_1 \\
y_{m+2} &= x_{m+2} \; op \; v_2 \\
&\ldots \\
y_{2m} &= x_{2m} \; op \; v_m \\
y_{2m+1} &= x_{2m+1} \; op \; v1 \\
y_{2m+2} &= x_{2m+2} \; op \; v2 \\
&\ldots
\end{aligned}$$

where $op = +, -, *, \text{ or } /$.

Examples of Usage

The multiplicative seasonal indices are 0.9, 1.3. 0.8, and 1.1 for the four quarters. Let ADJUST be a quarterly time series variable that has been seasonally adjusted in a multiplicative fashion. To restore the seasonality to ADJUST use the following transformation:

```
   convert adjust=seasonal / transformout=(seqmult (0.9 1.3 0.8 1.1));
```

The additive seasonal indices are 4.4, -1.1, -2.1, and -1.2 for the four quarters. Let ADJUST be a quarterly time series variable that has been seasonally adjusted in additive fashion. To restore the seasonality to ADJUST use the following transformation:

```
   convert adjust=seasonal / transformout=(seqadd (4.4 -1.1 -2.1 -1.2));
```

Set Operators

For the set operators, the first parameter, n_1, represents the value to be replaced and the second parameter, n_2, represents the replacement value. The replacement can be localized to the beginning, middle, or end of the series.

Examples of Usage

Suppose that a store opened recently and that the sales history is stored in a database that does not recognize missing values. Even though demand may have existed prior to the stores opening, this database assigns the value of zero. Modeling the sales history may be problematic because the sales history is mostly zero. To compensate for this deficiency, the leading zero values should be to missing with the remaining zero values unchanged (representing no demand).

```
   convert sales=demand / transformout=(setleft 0 .);
```

Likewise, suppose a store is closed recently. The demand may still be present and hence a recorded value of zero does not accurately reflect actual demand.

```
   convert sales=demand / transformout=(setright 0 .);
```

Scale Operator

For the scale operator, the first parameter, n_1, represents the value associated with the minimum value (y_{min}) and the second parameter, n_2, represents the value associated with the maximum value (y_{max}) of the original series (y_t). The scale operator scales the data between the parameters n_1 and n_2 as follows:

$$x_t = ((n_2 - n_1)/(y_{max} - y_{min}))(y_t - y_{min}) + n_1$$

Examples of Usage

Suppose that two new product sales histories are stored in variables X and Y and you wish to determine their adoption rates. In order to compare their adoption histories the variables must be scaled for comparison.

```
convert x=w / transformout=(scale 0 1);
convert y=z / transformout=(scale 0 1);
```

Adjust Operator

For the moving summation and product window operators, the window widths at the beginning and end of the series are smaller than those in the middle of the series. Likewise, if there are embedded missing values, the window width will be smaller than specified. When preceded by the ADJUST operator, the moving summation (MOVSUM CMOVSUM) and moving product operators (MOVPROD CMOVPROD) are adjusted by the window width.

For example, suppose the variable X has 10 values and the moving summation operator of width 3 is applied to X to create the variable Y with window width adjustment and the variable Z without adjustment.

```
convert x=y / transformout=(adjust movsum 3);
convert x=z / transformout=(movsum 3);
```

The above transformation result in the following: $y_1 = 3z_1$, $y_2 = \frac{3}{2}z_2$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

For example, suppose the variable X has 10 values and the moving multiplicative operator of width 3 is applied to X to create the variable Y with window width adjustment and the variable Z without adjustment.

```
convert x=y / transformout=(adjust movprod 3);
convert x=z / transformout=(movprod 3);
```

The above transformation result in the following: $y_1 = z_1^3$, $y_2 = z_2^{3/2}$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

Moving T-Value Operators

The moving *t*-value operators (CUTVALUE, MOVTVALUE, CMOVTVALUE) compute the *t*-value of the cumulative series or moving window. They can be viewed as combinations of the moving average (CUAVE, MOVAVE, CMOVAVE) and the moving standard deviation (CUSTD, MOVSTD, CMOVSTD), respectively.

Percent Operators

The percentage operators compute the percent summation and the percent difference of the current value and the $\text{lag}(n)$. The percent summation operator (PCTSUM) computes $y_t = 100x_t/\text{cusum}(x_{t-n})$. If any of the values of the preceding equation are missing or the cumulative summation is zero, the result is set to missing. The percent difference operator (PCTDIF) computes $y_t = 100(x_t - x_{t-n})/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose variable X contains the series. The percent summation of lag 4 is applied to X to create the variable Y. The percent difference of lag 4 is applied to X to create the variable Z.

```
convert x=y / transformout=(pctsum 4);
convert x=z / transformout=(pctdif 4);
```

Ratio Operators

The ratio operator computes the ratio of the current value and the $\text{lag}(n)$. The ratio operator (RATIO) computes $y_t = x_t/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose variable X contains the series. The ratio of lag 4 is applied to X to create the variable Y. The percent ratio of lag 4 is applied to X to create the variable Z.

```
convert x=y / transformout=(ratio 4);
convert x=z / transformout=(ratio 4 * 100);
```

OUT= Data Set

The OUT= output data set contains the following variables:

- the BY variables, if any
- an ID variable that identifies the time period for each output observation
- the result variables
- if no frequency conversion is performed (so that there is one output observation corresponding to each input observation), all the other variables in the input data set are copied to the output data set

The ID variable in the output data set is named as follows:

- If an ID statement is used, the new ID variable has the same name as the variable used in the ID statement.
- If no ID statement is used, but the FROM= option is used, then the name of the ID variable is either DATE or DATETIME, depending on whether the TO= option indicates SAS date or SAS datetime values.
- If neither an ID statement nor the TO= option is used, the ID variable is named TIME.

OUTEST= Data Set

The OUTEST= data set contains the coefficients of the spline curves fit to the input series. The OUTEST= data set is of interest if you want to verify the interpolating curve PROC EXPAND uses, or if you want to use this function in another context, (for example, in a SAS/IML program).

The OUTEST= data set contains the following variables:

- the BY variables, if any
- VARNAME, a character variable containing the name of the input variable to which the coefficients apply
- METHOD, a character variable containing the value of the METHOD= option used to fit the series
- OBSERVED, a character variable containing the first letter of the OBSERVED= option name for the input series
- the ID variable that contains the lower breakpoint (or “knot”) of the spline segment to which the coefficients apply. The ID variable has the same name as the variable used in the ID statement. If an ID statement is not used, but the FROM= option is used, then the name of the ID variable is DATE or DATETIME, depending on whether the FROM= option indicates SAS date or SAS datetime values. If neither an ID statement nor the FROM= option is used, the ID variable is named TIME.
- CONSTANT, the constant coefficient for the spline segment
- LINEAR, the linear coefficient for the spline segment
- QUAD, the quadratic coefficient for the spline segment
- CUBIC, the cubic coefficient for the spline segment

For each BY group, the OUTEST= data set contains observations for each polynomial segment of the spline curve fit to each input series. To obtain the observations defining the spline curve used for a series, select the observations where the value of VARNAME equals the name of the series.

The observations for a series in the OUTEST= data set encode the spline function fit to the series as follows. Let $a_i, b_i, c_i,$ and d_i be the values of the variables CUBIC, QUAD, LINEAR, and CONSTANT, respectively, for the *i*th observation for the series. Let x_i be the value of the ID variable for the *i*th observation for the series. Let *n* be the number of observations in the OUTEST= data set for the series. The value of the spline function evaluated at a point *x* is

$$f(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

where the segment number *i* is selected as follows:

$$i = \begin{cases} i & \text{such that } x_i \le x < x_{i+1}, 1 \le i < n \\ 1 & \text{if } x < x_1 \\ n & \text{if } x \ge x_n \end{cases}$$

In other words, if *x* is between the first and last ID values ($x_1 \le x < x_n$), use the observation from the OUTEST= data set with the largest ID value less than or equal to *x*. If *x* is less than the first ID value x_1, then $i = 1$. If *x* is greater than or equal to the last ID value ($x \ge x_n$), then $i = n$.

For METHOD=JOIN, the curve is a linear spline, and the values of CUBIC and QUAD are 0. For METHOD=STEP, the curve is a constant spline, and the values of CUBIC, QUAD, and LINEAR are 0. For METHOD=AGGREGATE, no coefficients are output.

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the EXPAND procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement and the PLOT= option in the EXPAND statement according to the following syntax. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

PLOT= *option* **|** **(***options***)**

specifies the graphical output desired. If the PLOT= option is used, the specified graphical output is produced for each output variable specified by a CONVERT statement. By default, the EXPAND procedure produces no graphical output. The following PLOT= options are available:

INPUT	plots the input series.
TRANSFORMIN	plots the transformed input series. (TRANSFORMIN= option)
CROSSINPUT	plots both the input series and the transformed input series. (TRANSFORMIN= option must be specified.)
JOINTINPUT	plots both the input series and the transformed input series. (TRANSFORMIN= option must be specified.)
CONVERTED|METHOD	plots the converted series. (METHOD= option)

TRANSFORMOUT	plots the transformed output series. (TRANSFORMOUT= option)
CROSSOUTPUT	plots both the converted series and the transformed output series. (TRANSFORMOUT= option must be specified.)
JOINTOUTPUT	plots both the converted series and the transformed output series. (TRANSFORMOUT= option must be specified.)
SERIES|OUTPUT	plots the series stored in the OUT= data set. (combination of TRANSFORMIN=, METHOD=, and/or TRANFORMOUT= options)
ALL	Same as PLOT=(INPUT TRANFORMIN CONVERTED TRANSFORMOUT).

The PLOT= option produces results associated with each CONVERT statement output variable and the options listed next to the above PLOT= options in parenthesis. The PLOT= option produces output for these results utilizing the Output Delivery System (ODS).

The PLOT=TRANSFORMIN plots the series after the input transformation (TRANSFORMIN= option) is applied. If the TRANFORMIN= option is not specified in the CONVERT statement for an output variable, the input transformation plot is not produced.

The PLOT=CROSSINPUT plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. The left side vertical axis refers to the input series, while the right side vertical axis refers to the series after the transformation. If the TRANFORMIN= option is not specified in the CONVERT statement for an output variable, the cross input plot is not produced.

The PLOT=JOINTINPUT jointly plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. If the TRANFORMIN= option is not specified in the CONVERT statement for an output variable, the joint input plot is not produced.

The PLOT=CONVERT plots the series after the input transformation (TRANSFORMIN= option) is applied and after frequency conversion (METHOD= option). If there is no frequency conversion for an output variable, the converted series plot is not produced.

The PLOT=TRANSFORMOUT plots the series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANFORMOUT= option is not specified in the CONVERT statement for an output variable, the output transformation plot is not produced.

The PLOT=CROSSOUTPUT plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. The left side vertical axis refers to the input series, while the right side vertical axis refers to the series after the transformation. If the TRANFORMOUT= option is not specified in the CONVERT statement for an output variable, the cross output plot is not produced.

The PLOT=JOINTOUTPUT jointly plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANFORMOUT= option is not specified in the CONVERT statement for an output variable, the joint output plot is not produced.

The PLOT=SERIES option plots the series after it has undergone input transformation (TRANSFORMIN= option), frequency conversion (METHOD= option), and output transformation (TRANSFORMOUT= option) if these CONVERT statement options were specified.

The PLOT=(ALL SERIES) option plots the series in the same way as the previous example as well as the intermediate series associated with the CONVERT statement options (TRANSFORMIN=, METHOD=, and/or TRANFORMOUT= options if specified).

The PLOT=(ALL SERIES JOINTINPUT JOINTOUTPUT CROSSINPUT CROSSOUTPUT) option plots the series in the same way as the previous example as well as the joint and cross plots series associated with the CONVERT statement options (TRANSFORMIN=, METHOD=, and/or TRANFORMOUT= options if specified).

Note: The joint graphics options (PLOT=JOINTINPUT or PLOT=JOINTOUTPUT) plot the (input or converted) series and the transformed series on the same scale. If the transformation changes the range of the (input or converted) series, these plots may be hard to visualize.

ODS Graph Names

PROC EXPAND assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 16.2.

To request these graphs, you must specify the ODS GRAPHICS statement and the PLOT= option in the EXPAND statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 16.2. ODS Graphics Produced by PROC EXPAND

ODS Graph Name	Plot Description	PLOT= Option Argument
ConvertedSeriesPlot	Converted Series Plot	CONVERTED|METHOD|SERIES|OUTPUT|ALL
CrossInputSeriesPlot	Cross Input Series Plot	CROSSINPUT
CrossOutputSeriesPlot	Cross Output Series Plot	CROSSOUTPUT
InputSeriesPlot	Input Series Plot	INPUT|JOINTINPUT|ALL
JointInputSeriesPlot	Joint Input Series Plot	JOINTINPUT
JointOutputSeriesPlot	Joint Output Series Plot	JOINTOUTPUT
OutputSeriesPlot	Output Series Plot	SERIES|OUTPUT

Table 16.2. (continued)

ODS Graph Name	Plot Description	PLOT= Option Argument
TransformedInputSeriesPlot	Transformed Input Series Plot	TRANSFORMIN|SERIES|OUTPUT|ALL
TransformedOutputSeriesPlot	Transformed Output Series Plot	TRANSFORMOUT|SERIES|OUTPUT|ALL

Examples

Example 16.1. Combining Monthly and Quarterly Data

This example combines monthly and quarterly data sets by interpolating monthly values for the quarterly series. The series are extracted from two small sample data sets stored in the SASHELP library. These data sets were contributed by Citicorp Data Base services and contain selected U.S. macro economic series.

The quarterly series gross domestic product (GDP) and implicit price deflator (GD) are extracted from SASHELP.CITIQTR. The monthly series industrial production index (IP) and unemployment rate (LHUR) are extracted from SASHELP.CITIMON. Only observations for the years 1990 and 1991 are selected. PROC EXPAND is then used to interpolate monthly estimates for the quarterly series, and the interpolated series are merged with the monthly data.

The following statements extract and print the quarterly data, shown in Output 16.1.1.

```
data qtrly;
   set sashelp.citiqtr;
   where date >= '1jan1990'd &
         date <  '1jan1992'd ;
   keep date gdp gd;
run;

title "Quarterly Data";
proc print data=qtrly;
run;
```

Output 16.1.1. Quarterly Data Set

```
                                  Quarterly Data

                     Obs      DATE        GD          GDP

                      1     1990:1     111.100     5422.40
                      2     1990:2     112.300     5504.70
                      3     1990:3     113.600     5570.50
                      4     1990:4     114.500     5557.50
                      5     1991:1     115.900     5589.00
                      6     1991:2     116.800     5652.60
                      7     1991:3     117.400     5709.20
                      8     1991:4        .        5736.60
```

The following statements extract and print the monthly data, shown in Output 16.1.2.

```
data monthly;
   set sashelp.citimon;
   where date >= '1jan1990'd &
         date <  '1jan1992'd ;
   keep date ip lhur;
run;

title "Monthly Data";
proc print data=monthly;
run;
```

Output 16.1.2. Monthly Data Set

```
                           Monthly Data

                  Obs       DATE        IP        LHUR

                    1    JAN1990    107.500    5.30000
                    2    FEB1990    108.500    5.30000
                    3    MAR1990    108.900    5.20000
                    4    APR1990    108.800    5.40000
                    5    MAY1990    109.400    5.30000
                    6    JUN1990    110.100    5.20000
                    7    JUL1990    110.400    5.40000
                    8    AUG1990    110.500    5.60000
                    9    SEP1990    110.600    5.70000
                   10    OCT1990    109.900    5.80000
                   11    NOV1990    108.300    6.00000
                   12    DEC1990    107.200    6.10000
                   13    JAN1991    106.600    6.20000
                   14    FEB1991    105.700    6.50000
                   15    MAR1991    105.000    6.70000
                   16    APR1991    105.500    6.60000
                   17    MAY1991    106.400    6.80000
                   18    JUN1991    107.300    6.90000
                   19    JUL1991    108.100    6.80000
                   20    AUG1991    108.000    6.80000
                   21    SEP1991    108.400    6.80000
                   22    OCT1991    108.200    6.90000
                   23    NOV1991    108.000    6.90000
                   24    DEC1991    107.800    7.10000
```

The following statements interpolate monthly estimates for the quarterly series and merge the interpolated series with the monthly data. The resulting combined data set is then printed, as shown in Output 16.1.3.

```
proc expand data=qtrly out=temp from=qtr to=month;
   convert gdp gd / observed=average;
   id date;
run;

data combined;
   merge monthly temp;
   by date;
run;

title "Combined Data Set";
proc print data=combined;
run;
```

Output 16.1.3. Combined Data Set

```
                          Combined Data Set

         Obs        DATE        IP         LHUR        GDP          GD

           1     JAN1990     107.500     5.30000     5409.69     110.879
           2     FEB1990     108.500     5.30000     5417.67     111.048
           3     MAR1990     108.900     5.20000     5439.39     111.367
           4     APR1990     108.800     5.40000     5470.58     111.802
           5     MAY1990     109.400     5.30000     5505.35     112.297
           6     JUN1990     110.100     5.20000     5538.14     112.801
           7     JUL1990     110.400     5.40000     5563.38     113.264
           8     AUG1990     110.500     5.60000     5575.69     113.641
           9     SEP1990     110.600     5.70000     5572.49     113.905
          10     OCT1990     109.900     5.80000     5561.64     114.139
          11     NOV1990     108.300     6.00000     5553.83     114.451
          12     DEC1990     107.200     6.10000     5556.92     114.909
          13     JAN1991     106.600     6.20000     5570.06     115.452
          14     FEB1991     105.700     6.50000     5588.18     115.937
          15     MAR1991     105.000     6.70000     5608.68     116.314
          16     APR1991     105.500     6.60000     5630.81     116.600
          17     MAY1991     106.400     6.80000     5652.92     116.812
          18     JUN1991     107.300     6.90000     5674.06     116.988
          19     JUL1991     108.100     6.80000     5693.43     117.164
          20     AUG1991     108.000     6.80000     5710.54     117.380
          21     SEP1991     108.400     6.80000     5724.11     117.665
          22     OCT1991     108.200     6.90000     5733.65        .
          23     NOV1991     108.000     6.90000     5738.46        .
          24     DEC1991     107.800     7.10000     5737.75        .
```

Example 16.2. Interpolating Irregular Observations

This example shows the interpolation of a series of values measured at irregular points in time. The data are hypothetical. Assume that a series of randomly timed quality control inspections are made and defect rates for a process are measured. The problem is to produce two reports: estimates of monthly average defect rates for the months within the period covered by the samples, and a plot of the interpolated defect rate curve over time.

The following statements read and print the input data, as shown in Output 16.2.1.

```
data samples;
  input date : date9. defects @@;
  label defects = "Defects per 1000 units";
  format date date9.;
datalines;
13jan1992    55    27jan1992   73    19feb1992   84    8mar1992   69
27mar1992    66     5apr1992   77    29apr1992   63   11may1992   81
25may1992    89     7jun1992   94    23jun1992  105   11jul1992   97
15aug1992   112    29aug1992   89    10sep1992   77   27sep1992   82
;

title "Sampled Defect Rates";
proc print data=samples;
run;
```

Output 16.2.1. Measured Defect Rates

```
                         Sampled Defect Rates

                     Obs         date    defects

                       1    13JAN1992       55
                       2    27JAN1992       73
                       3    19FEB1992       84
                       4    08MAR1992       69
                       5    27MAR1992       66
                       6    05APR1992       77
                       7    29APR1992       63
                       8    11MAY1992       81
                       9    25MAY1992       89
                      10    07JUN1992       94
                      11    23JUN1992      105
                      12    11JUL1992       97
                      13    15AUG1992      112
                      14    29AUG1992       89
                      15    10SEP1992       77
                      16    27SEP1992       82
```

To compute the monthly estimates, use PROC EXPAND with the TO=MONTH option and specify OBSERVED=(BEGINNING,AVERAGE). The following statements interpolate the monthly estimates.

```
proc expand data=samples out=monthly to=month;
  id date;
  convert defects / observed=(beginning,average);
run;

title "Estimated Monthly Average Defect Rates";
proc print data=monthly;
run;
```

The results are printed in Output 16.2.2.

Output 16.2.2. Monthly Average Estimates

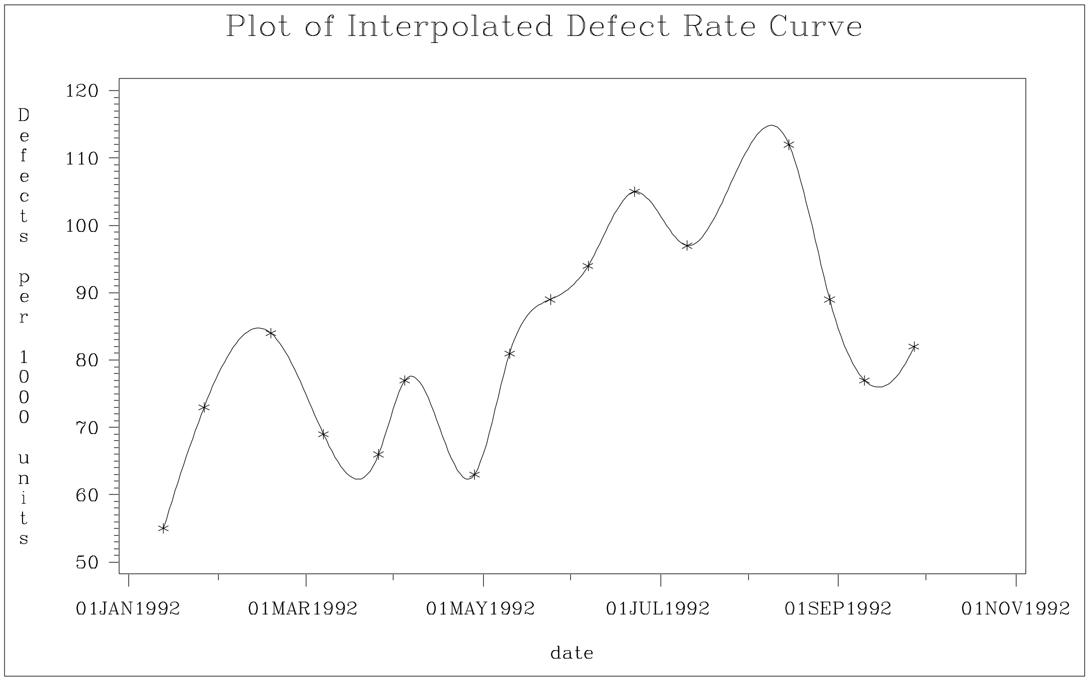
```
                 Estimated Monthly Average Defect Rates

                      Obs       date    defects

                        1     JAN1992     59.323
                        2     FEB1992     82.000
                        3     MAR1992     66.909
                        4     APR1992     70.205
                        5     MAY1992     82.762
                        6     JUN1992     99.701
                        7     JUL1992    101.564
                        8     AUG1992    105.491
                        9     SEP1992     79.206
```

To produce the plot, first use PROC EXPAND with TO=DAY to interpolate a full set of daily values, naming the interpolated series INTERPOL. Then merge this data set with the samples so you can plot both the measured and the interpolated values on the same graph. PROC GPLOT is used to plot the curve. The actual sample points are plotted with asterisks. The following statements interpolate and plot the defects rate curve.

```
proc expand data=samples out=daily to=day;
  id date;
  convert defects = interpol;
run;

data daily;
  merge daily samples;
  by date;
run;

title "Plot of Interpolated Defect Rate Curve";
proc gplot data=daily;
   axis2 label=(a=-90 r=90 );
   symbol1 v=none i=join;
   symbol2 v=star i=none;
   plot interpol * date = 1 defects * date = 2  /
        vaxis=axis2 overlay;
run;
quit;
```

The plot is shown in Output 16.2.3.

Output 16.2.3. Interpolated Defects Rate Curve



Example 16.3. Using Transformations

This example shows the use of PROC EXPAND to perform various transformations of time series. The following statements read in monthly values for a variable X.

```
data test;
   input year qtr x;
   date = yyq( year, qtr );
   format date yyqc.;
datalines;
1989 3 5238
1989 4 5289
1990 1 5375
1990 2 5443
1990 3 5514
1990 4 5527
1991 1 5557
1991 2 5615
;
```

The following statements use PROC EXPAND to compute lags and leads and a 3-period moving average of the X series.

```
proc expand data=test out=out method=none;
   id date;
   convert x = x_lag2   / transformout=(lag 2);
   convert x = x_lag1   / transformout=(lag 1);
   convert x;
   convert x = x_lead1  / transformout=(lead 1);
   convert x = x_lead2  / transformout=(lead 2);
   convert x = x_movave / transformout=(movave 3);
run;

title "Transformed Series";
proc print data=out;
run;
```

Because there are no missing values to interpolate and no frequency conversion, the METHOD=NONE option is used to prevent PROC EXPAND from performing unnecessary computations. Because no frequency conversion is done, all variables in the input data set are copied to the output data set. The CONVERT X; statement is included to control the position of X in the output data set. This statement can be omitted, in which case X is copied to the output data set following the new variables computed by PROC EXPAND.

The results are shown in Output 16.3.1.

Output 16.3.1. Output Data Set with Transformed Variables

```
                                Transformed Series

   Obs    date    x_lag2  x_lag1    x    x_lead1  x_lead2  x_movave   year   qtr

    1    1989:3      .       .     5238    5289     5375    5238.00   1989    3
    2    1989:4      .     5238    5289    5375     5443    5263.50   1989    4
    3    1990:1    5238    5289    5375    5443     5514    5300.67   1990    1
    4    1990:2    5289    5375    5443    5514     5527    5369.00   1990    2
    5    1990:3    5375    5443    5514    5527     5557    5444.00   1990    3
    6    1990:4    5443    5514    5527    5557     5615    5494.67   1990    4
    7    1991:1    5514    5527    5557    5615       .     5532.67   1991    1
    8    1991:2    5527    5557    5615      .        .     5566.33   1991    2
```

Example 16.4. Illustration of ODS Graphics (Experimental)

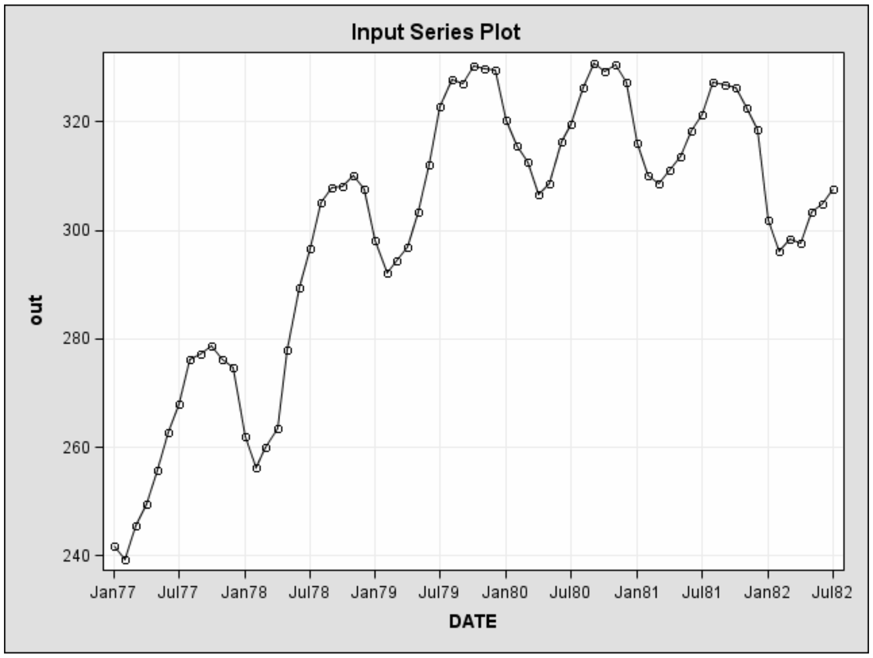
This example illustrates the use of experimental ODS graphics.

The graphical displays are requested by specifying the experimental ODS GRAPHICS statement and the experimental PLOT= option in the PROC EXPAND statement. For general information about ODS graphics, see Chapter 9, “Statistical Graphics Using ODS.” For specific information about the graphics available in the EXPAND procedure, see the “ODS Graphics” section on page 820.

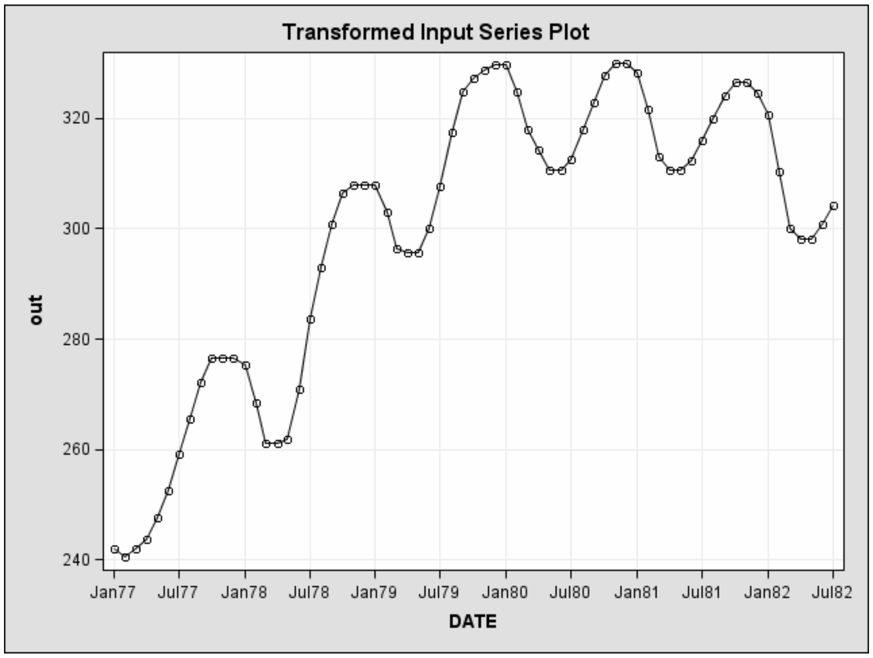
The following statements utilize the SASHELP.WORKERS data set to convert the time series of electrical workers from monthly to quarterly frequency, and display ODS graphics plots. The PLOT=ALL option is specified to request the plots of the input series, the transformed input series, the converted series, and the transformed output series. Output 16.4.1 through Output 16.4.4 show these plots.

```
ods html;
ods graphics on;

proc expand data=sashelp.workers out=out from=month to=qtr plot=all;
   id date;
   convert electric=out / transformin=(movmed 4) method=spline
                          transformout=(movave 3);
run;

ods graphics off;
ods html close;
```

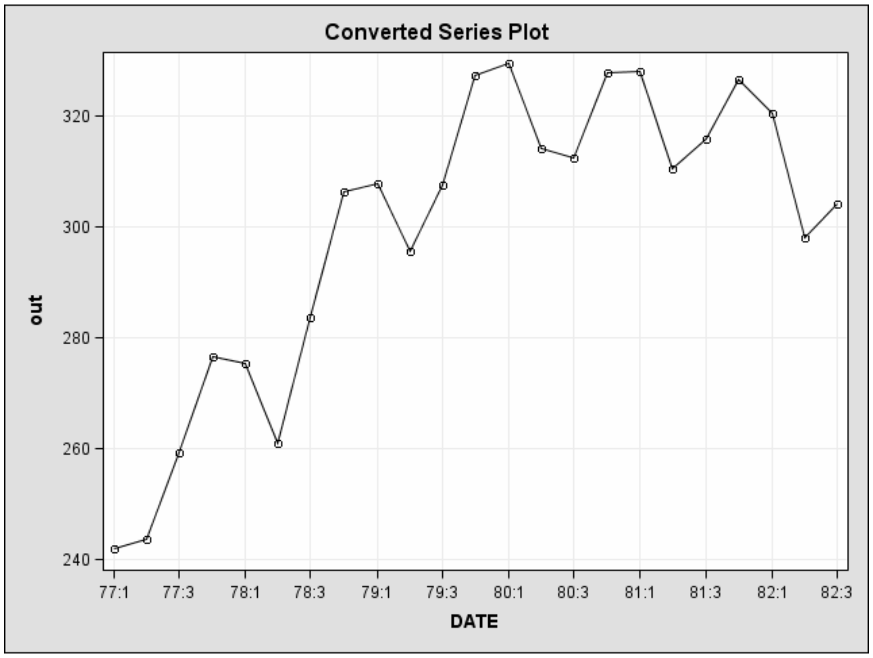
Output 16.4.1. Input Series Plot (Experimental)



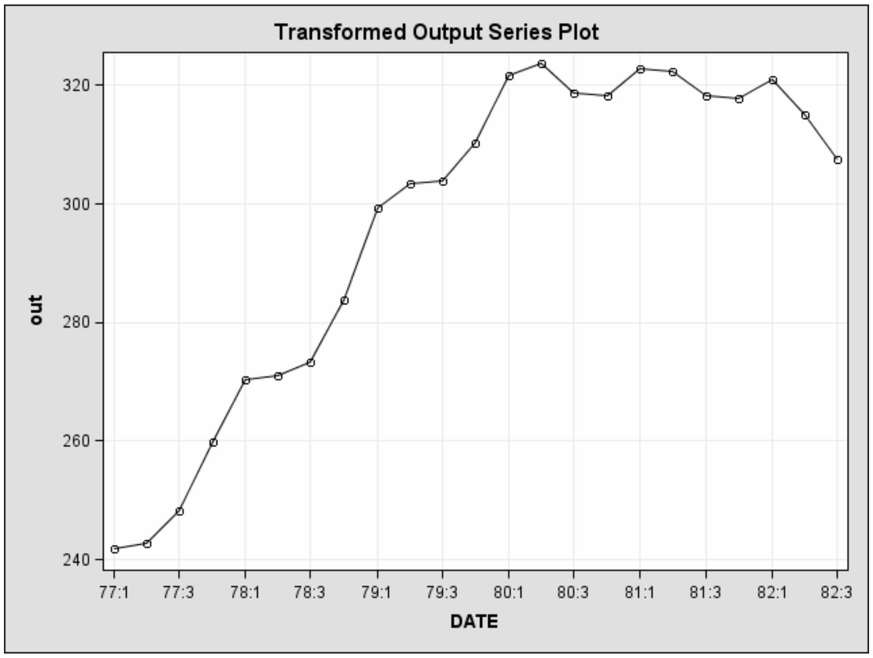
Output 16.4.2. Transformed Input Series Plot (Experimental)



Output 16.4.3. Converted Series Plot (Experimental)



Output 16.4.4. Transformed Output Series Plot (Experimental)



References

DeBoor, Carl (1981), *A Practical Guide to Splines*, New York: Springer-Verlag.

Hodrick, R. J., and Prescott, E. C. (1980). “Post-war U.S. business cycles: An empirical investigation.” Discussion paper 451, Carnegie-Mellon University.

Levenbach, H. and Cleary, J.P. (1984), *The Modern Forecaster*, Belmont, CA: Lifetime Learning Publications (a division of Wadsworth, Inc.), 129-133.

Makridakis, S. and Wheelwright, S.C. (1978), *Interactive Forecasting: Univariate and Multivariate Methods*, Second Edition, San Francisco: Holden-Day, 198-201.

Wheelwright, S.C. and Makridakis, S. (1973), *Forecasting Methods for Management*, Third Edition, New York: Whiley-Interscience, 123-133.

Chapter 17
The FORECAST Procedure

Chapter Contents

Chapter 17
The FORECAST Procedure

Overview

The FORECAST procedure provides a quick and automatic way to generate forecasts for many time series in one step. The procedure can forecast hundreds of series at a time, with the series organized into separate variables or across BY groups. PROC FORECAST uses extrapolative forecasting methods where the forecasts for a series are functions only of time and past values of the series, not of other variables.

You can use the following forecasting methods. For each of these methods, you can specify linear, quadratic, or no trend.

- The stepwise autoregressive method is used by default. This method combines time trend regression with an autoregressive model and uses a stepwise method to select the lags to use for the autoregressive process.
- The exponential smoothing method produces a time trend forecast, but in fitting the trend, the parameters are allowed to change gradually over time, and earlier observations are given exponentially declining weights. Single, double, and triple exponential smoothing are supported, depending on whether no trend, linear trend, or quadratic trend is specified. Holt two-parameter linear exponential smoothing is supported as a special case of the Holt-Winters method without seasons.
- The Winters method (also called Holt-Winters) combines a time trend with multiplicative seasonal factors to account for regular seasonal fluctuations in a series. Like the exponential smoothing method, the Winters method allows the parameters to change gradually over time, with earlier observations given exponentially declining weights. You can also specify the additive version of the Winters method, which uses additive instead of multiplicative seasonal factors. When seasonal factors are omitted, the Winters method reduces to the Holt two-parameter version of double exponential smoothing.

The FORECAST procedure writes the forecasts and confidence limits to an output data set, and can write parameter estimates and fit statistics to an output data set. The FORECAST procedure does not produce printed output.

PROC FORECAST is an extrapolation procedure useful for producing practical results efficiently. However, in the interest of speed, PROC FORECAST uses some shortcuts that cause some statistical results (such as confidence limits) to be only approximate. For many time series, the FORECAST procedure, with appropriately chosen methods and weights, can yield satisfactory results. Other SAS/ETS procedures can produce better forecasts but at greater computational expense.

You can perform the stepwise autoregressive forecasting method with the AUTOREG procedure. You can perform exponential smoothing with statistically optimal weights as an ARIMA model using the ARIMA procedure. Seasonal ARIMA models can be used for forecasting seasonal series for which the Winters and additive Winters methods might be used.

Additionally, the Time Series Forecasting System can be used to develop forecasting models, estimate the model parameters, evaluate the models' ability to forecast and display the results graphically. See Chapter 34, "Getting Started with Time Series Forecasting," for more details.

Getting Started

To use PROC FORECAST, specify the input and output data sets and the number of periods to forecast in the PROC FORECAST statement, then list the variables to forecast in a VAR statement.

For example, suppose you have monthly data on the sales of some product, in a data set, named PAST, as shown in Figure 17.1, and you want to forecast sales for the next 10 months.

```
                                 Obs     date      sales

                                   1    JUL89     9.5161
                                   2    AUG89     9.6994
                                   3    SEP89     9.2644
                                   4    OCT89     9.6837
                                   5    NOV89    10.0784
                                   6    DEC89     9.9005
                                   7    JAN90    10.2375
                                   8    FEB90    10.6940
                                   9    MAR90    10.6290
                                  10    APR90    11.0332
                                  11    MAY90    11.0270
                                  12    JUN90    11.4165
                                  13    JUL90    11.2918
                                  14    AUG90    11.3475
                                  15    SEP90    11.2913
                                  16    OCT90    11.3771
                                  17    NOV90    11.5457
                                  18    DEC90    11.6433
                                  19    JAN91    11.9293
                                  20    FEB91    11.9752
                                  21    MAR91    11.9283
                                  22    APR91    11.8985
                                  23    MAY91    12.0419
                                  24    JUN91    12.3537
                                  25    JUL91    12.4546
```

Figure 17.1. Example Data Set PAST

The following statements forecast 10 observations for the variable SALES using the default STEPAR method and write the results to the output data set PRED:

```
   proc forecast data=past lead=10 out=pred;
```

```
      var sales;
   run;
```

The following statements use the PRINT procedure to print the data set PRED:

```
   proc print data=pred;
   run;
```

The PROC PRINT listing of the forecast data set PRED is shown in Figure 17.2.

```
                    Obs     _TYPE_      _LEAD_      sales

                      1    FORECAST        1       12.6205
                      2    FORECAST        2       12.7665
                      3    FORECAST        3       12.9020
                      4    FORECAST        4       13.0322
                      5    FORECAST        5       13.1595
                      6    FORECAST        6       13.2854
                      7    FORECAST        7       13.4105
                      8    FORECAST        8       13.5351
                      9    FORECAST        9       13.6596
                     10    FORECAST       10       13.7840
```

Figure 17.2. Forecast Data Set PRED

Giving Dates to Forecast Values

Normally, your input data set has an ID variable that gives dates to the observations, and you want the forecast observations to have dates also. Usually, the ID variable has SAS date values. (See Chapter 2, “Working with Time Series Data,” for information on using SAS date values.) The ID statement specifies the identifying variable.

If the ID variable contains SAS date values, the INTERVAL= option should be used on the PROC FORECAST statement to specify the time interval between observations. (See Chapter 3, “Date Intervals, Formats, and Functions,” for more information on time intervals.) The FORECAST procedure uses the INTERVAL= option to generate correct dates for forecast observations.

The data set PAST, shown in Figure 17.1, has monthly observations and contains an ID variable DATE with SAS date values identifying each observation. The following statements produce the same forecast as the preceding example and also include the ID variable DATE in the output data set. Monthly SAS date values are extrapolated for the forecast observations.

```
   proc forecast data=past interval=month lead=10 out=pred;
      var sales;
      id date;
   run;
```

Computing Confidence Limits

Depending on the output options specified, multiple observations are written to the OUT= data set for each time period. The different parts of the results are contained in the VAR statement variables in observations identified by the character variable _TYPE_ and by the ID variable.

For example, the following statements use the OUTLIMIT option to write forecasts and 95% confidence limits for the variable SALES to the output data set PRED. This data set is printed with the PRINT procedure.

```
proc forecast data=past interval=month lead=10
              out=pred outlimit;
   var sales;
   id date;
run;

proc print data=pred;
run;
```

The output data set PRED is shown in Figure 17.3.

```
             Obs    date     _TYPE_      _LEAD_     sales

               1    AUG91    FORECAST       1      12.6205
               2    AUG91    L95            1      12.1848
               3    AUG91    U95            1      13.0562
               4    SEP91    FORECAST       2      12.7665
               5    SEP91    L95            2      12.2808
               6    SEP91    U95            2      13.2522
               7    OCT91    FORECAST       3      12.9020
               8    OCT91    L95            3      12.4001
               9    OCT91    U95            3      13.4039
              10    NOV91    FORECAST       4      13.0322
              11    NOV91    L95            4      12.5223
              12    NOV91    U95            4      13.5421
              13    DEC91    FORECAST       5      13.1595
              14    DEC91    L95            5      12.6435
              15    DEC91    U95            5      13.6755
              16    JAN92    FORECAST       6      13.2854
              17    JAN92    L95            6      12.7637
              18    JAN92    U95            6      13.8070
              19    FEB92    FORECAST       7      13.4105
              20    FEB92    L95            7      12.8830
              21    FEB92    U95            7      13.9379
              22    MAR92    FORECAST       8      13.5351
              23    MAR92    L95            8      13.0017
              24    MAR92    U95            8      14.0686
              25    APR92    FORECAST       9      13.6596
              26    APR92    L95            9      13.1200
              27    APR92    U95            9      14.1993
              28    MAY92    FORECAST      10      13.7840
              29    MAY92    L95           10      13.2380
              30    MAY92    U95           10      14.3301
```

Figure 17.3. Output Data Set

Form of the OUT= Data Set

The OUT= data set PRED, shown in Figure 17.3, contains three observations for each of the 10 forecast periods. Each of these three observations has the same value of the ID variable DATE, the SAS date value for the month and year of the forecast.

The three observations for each forecast period have different values of the variable _TYPE_. For the _TYPE_=FORECAST observation, the value of the variable SALES is the forecast value for the period indicated by the DATE value. For the _TYPE_=L95 observation, the value of the variable SALES is the lower limit of the 95% confidence interval for the forecast. For the _TYPE_=U95 observation, the value of the variable SALES is the upper limit of the 95% confidence interval.

You can control the types of observations written to the OUT= data set with the PROC FORECAST statement options OUTLIMIT, OUTRESID, OUTACTUAL, OUT1STEP, OUTSTD, OUTFULL, and OUTALL. For example, the OUTFULL option outputs the confidence limit values, the one-step-ahead predictions, and the actual data, in addition to the forecast values. See the sections "Syntax" and "OUT= Data Set" later in this chapter for more information.

Plotting Forecasts

The forecasts, confidence limits, and actual values can be plotted on the same graph with the GPLOT procedure. Use the appropriate output control options on the PROC FORECAST statement to include in the OUT= data set the series you want to plot. Use the _TYPE_ variable in the GPLOT procedure PLOT statement to separate the observations for the different plots.

In this example, the OUTFULL option is used, and the resulting output data set contains the actual and predicted values, as well as the upper and lower 95

```
proc forecast data=past interval=month lead=10
              out=pred outfull;
   id date;
   var sales;
run;

proc gplot data=pred;
   plot sales * date = _type_ /
        haxis= '1jan90'd to '1jan93'd by qtr
        href='15jul91'd;
   symbol1 i=none   v=star; /* for _type_=ACTUAL */
   symbol2 i=spline v=circle;   /* for _type_=FORECAST */
   symbol3 i=spline l=3;        /* for _type_=L95 */
   symbol4 i=spline l=3;        /* for _type_=U95 */
   where date >= '1jan90'd;
run;
```

The _TYPE_ variable is used in the GPLOT procedure's PLOT statement to make separate plots over time for each type of value. A reference line marks the start of the forecast period. (Refer to *SAS/GRAPH Software: Reference, Volume 2, Version 7, First Edition* for more information on using PROC GPLOT.) The WHERE statement restricts the range of the actual data shown in the plot. In this example, the variable SALES has monthly data from July 1989 through July 1991, but only the data for 1990 and 1991 are shown in the plot.

The plot is shown in Figure 17.4.

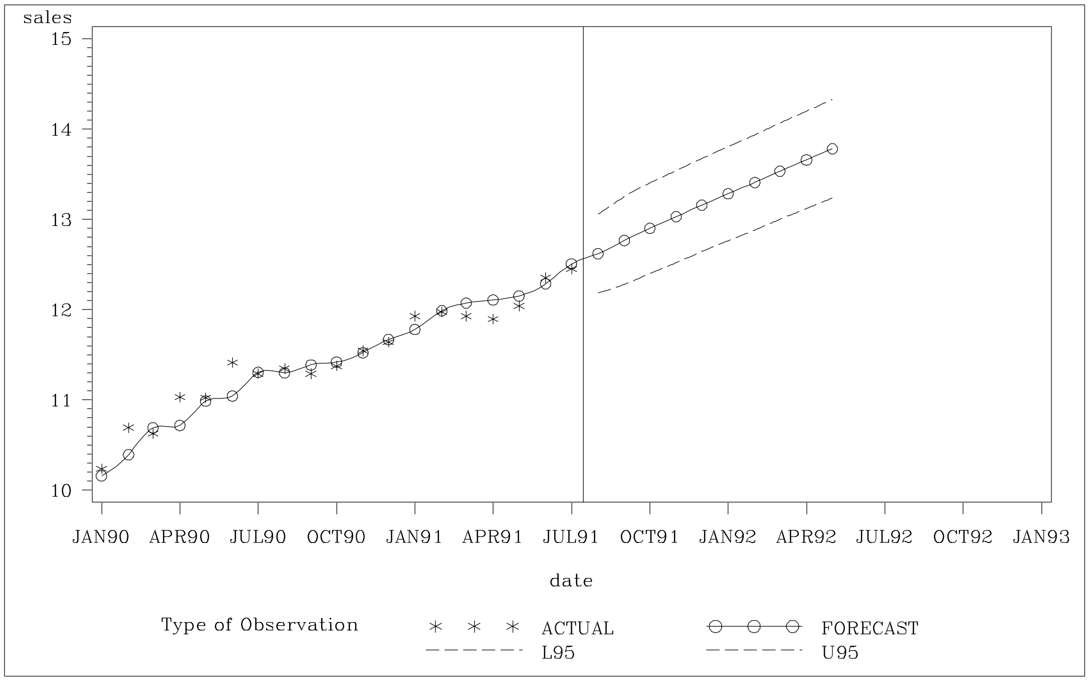


Figure 17.4. Plot of Forecast with Confidence Limits

Plotting Residuals

You can plot the residuals from the forecasting model using PROC GPLOT and a WHERE statement.

1. Use the OUTRESID option or the OUTALL option in the PROC FORECAST statement to include the residuals in the output data set.
2. Use a WHERE statement to specify the observation type of ’RESIDUAL’ in the PROC GPLOT code.

The following example adds the OUTRESID option to the preceding example and plots the residuals:

```
proc forecast data=past interval=month lead=10
              out=pred outfull outresid;
   id date;
   var sales;
run;

proc gplot data=pred;
```

```
      where _type_='RESIDUAL';
      plot sales * date /
           haxis= '1jan89'd to '1oct91'd by qtr;
      symbol1 i=needle v=circle;
   run;
```

The plot of residuals is shown in Figure 17.5.

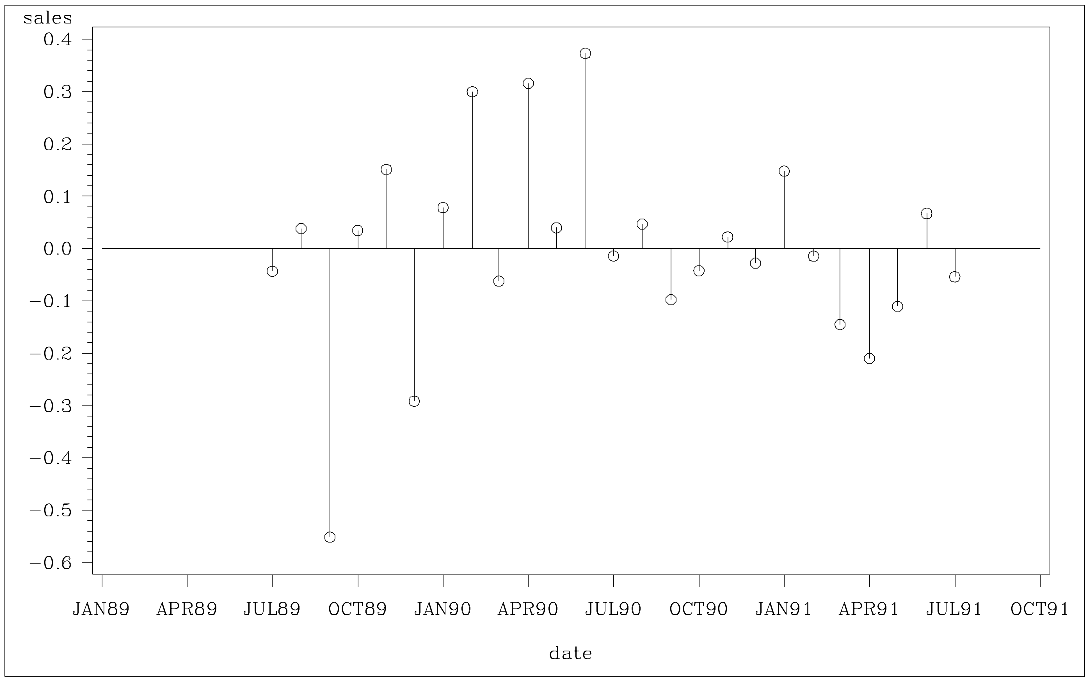


Figure 17.5. Plot of Residuals

Model Parameters and Goodness-of-Fit Statistics

You can write the parameters of the forecasting models used, as well as statistics measuring how well the forecasting models fit the data, to an output SAS data set using the OUTEST= option. The options OUTFITSTATS, OUTESTTHEIL, and OUTESTALL control what goodness-of-fit statistics are added to the OUTEST= data set.

For example, the following statements add the OUTEST= and OUTFITSTATS options to the previous example to create the output statistics data set EST for the results of the default stepwise autoregressive forecasting method:

```
   proc forecast data=past interval=month lead=10
                 out=pred outfull outresid
                 outest=est outfitstats;
      id date;
      var sales;
   run;
```

```
proc print data=est;
run;
```

The PRINT procedure prints the OUTEST= data set, as shown in Figure 17.6.

```
                      Obs    _TYPE_       date        sales

                        1    N           JUL91           25
                        2    NRESID      JUL91           25
                        3    DF          JUL91           22
                        4    SIGMA       JUL91    0.2001613
                        5    CONSTANT    JUL91    9.4348822
                        6    LINEAR      JUL91    0.1242648
                        7    AR1         JUL91    0.5206294
                        8    AR2         JUL91            .
                        9    AR3         JUL91            .
                       10    AR4         JUL91            .
                       11    AR5         JUL91            .
                       12    AR6         JUL91            .
                       13    AR7         JUL91            .
                       14    AR8         JUL91            .
                       15    SST         JUL91     21.28342
                       16    SSE         JUL91    0.8793714
                       17    MSE         JUL91    0.0399714
                       18    RMSE        JUL91    0.1999286
                       19    MAPE        JUL91    1.2280089
                       20    MPE         JUL91    -0.050139
                       21    MAE         JUL91    0.1312115
                       22    ME          JUL91    -0.001811
                       23    MAXE        JUL91    0.3732328
                       24    MINE        JUL91    -0.551605
                       25    MAXPE       JUL91    3.2692294
                       26    MINPE       JUL91    -5.954022
                       27    RSQUARE     JUL91    0.9586828
                       28    ADJRSQ      JUL91    0.9549267
                       29    RW_RSQ      JUL91    0.2657801
                       30    ARSQ        JUL91    0.9474145
                       31    APC         JUL91     0.044768
                       32    AIC         JUL91    -77.68559
                       33    SBC         JUL91    -74.02897
                       34    CORR        JUL91    0.9791313
```

Figure 17.6. The OUTEST= Data Set for STEPAR Method

In the OUTEST= data set, the DATE variable contains the ID value of the last observation in the data set used to fit the forecasting model. The variable SALES contains the statistic indicated by the value of the _TYPE_ variable. The _TYPE_=N, NRESID, and DF observations contain, respectively, the number of observations read from the data set, the number of nonmissing residuals used to compute the goodness-of-fit statistics, and the number of nonmissing observations minus the number of parameters used in the forecasting model.

The observation having _TYPE_=SIGMA contains the estimate of the standard deviation of the one-step prediction error computed from the residuals. The _TYPE_=CONSTANT and _TYPE_=LINEAR contain the coefficients of the time trend regression. The _TYPE_=AR1, AR2, ..., AR8 observations contain the estimated autoregressive parameters. A missing autoregressive parameter indicates that the autoregressive term at that lag was not included in the model by the stepwise model selection method. (See the section "STEPAR Method" later in this chapter for more information.)

The other observations in the OUTEST= data set contain various goodness-of-fit statistics that measure how well the forecasting model used fits the given data. See "OUTEST= Data Set" later in this chapter for details.

Controlling the Forecasting Method

The METHOD= option controls which forecasting method is used. The TREND= option controls the degree of the time trend model used. For example, the following statements produce forecasts of SALES as in the preceding example but use the double exponential smoothing method instead of the default STEPAR method:

```
proc forecast data=past interval=month lead=10
              method=expo trend=2
              out=pred outfull outresid
              outest=est outfitstats;
   var sales;
   id date;
run;

proc print data=est;
run;
```

The PRINT procedure prints the OUTEST= data set for the EXPO method, as shown in Figure 17.7.

```
                     Obs    _TYPE_       date        sales

                       1    N           JUL91           25
                       2    NRESID      JUL91           25
                       3    DF          JUL91           23
                       4    WEIGHT      JUL91    0.1055728
                       5    S1          JUL91    11.427657
                       6    S2          JUL91    10.316473
                       7    SIGMA       JUL91    0.2545069
                       8    CONSTANT    JUL91    12.538841
                       9    LINEAR      JUL91    0.1311574
                      10    SST         JUL91     21.28342
                      11    SSE         JUL91    1.4897965
                      12    MSE         JUL91    0.0647738
                      13    RMSE        JUL91    0.2545069
                      14    MAPE        JUL91    1.9121204
                      15    MPE         JUL91    -0.816886
                      16    MAE         JUL91    0.2101358
                      17    ME          JUL91    -0.094941
                      18    MAXE        JUL91    0.3127332
                      19    MINE        JUL91    -0.460207
                      20    MAXPE       JUL91    2.9243781
                      21    MINPE       JUL91    -4.967478
                      22    RSQUARE     JUL91     0.930002
                      23    ADJRSQ      JUL91    0.9269586
                      24    RW_RSQ      JUL91    -0.243886
                      25    ARSQ        JUL91    0.9178285
                      26    APC         JUL91    0.0699557
                      27    AIC         JUL91    -66.50591
                      28    SBC         JUL91    -64.06816
                      29    CORR        JUL91    0.9772418
```

Figure 17.7. The OUTEST= Data Set for METHOD=EXPO

See the "Syntax" section later in this chapter for other options that control the forecasting method. See "Introduction to Forecasting Methods" and "Forecasting Methods" later in this chapter for an explanation of the different forecasting methods.

Introduction to Forecasting Methods

This section briefly introduces the forecasting methods used by the FORECAST procedure. Refer to textbooks on forecasting and see "Forecasting Methods" later in this chapter for more detailed discussions of forecasting methods.

The FORECAST procedure combines three basic models to fit time series:

- time trend models for long-term, deterministic change
- autoregressive models for short-term fluctuations
- seasonal models for regular seasonal fluctuations

Two approaches to time series modeling and forecasting are *time trend models* and *time series methods*.

Time Trend Models

Time trend models assume that there is some permanent deterministic pattern across time. These models are best suited to data that are not dominated by random fluctuations.

Examining a graphical plot of the time series you want to forecast is often very useful in choosing an appropriate model. The simplest case of a time trend model is one in which you assume the series is a constant plus purely random fluctuations that are independent from one time period to the next. Figure 17.8 shows how such a time series might look.

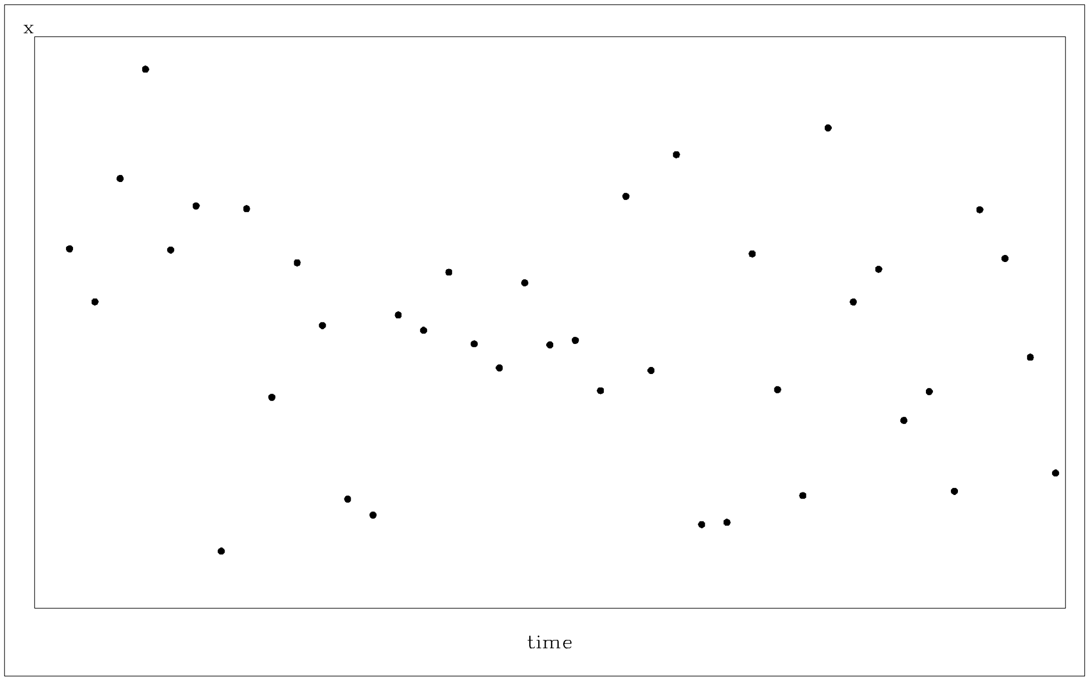


Figure 17.8. Time Series without Trend

The x_t values are generated according to the equation

$$x_t = b_0 + \epsilon_t$$

where ϵ_t is an independent, zero-mean, random error, and b_0 is the true series mean.

Suppose that the series exhibits growth over time, as shown in Figure 17.9.

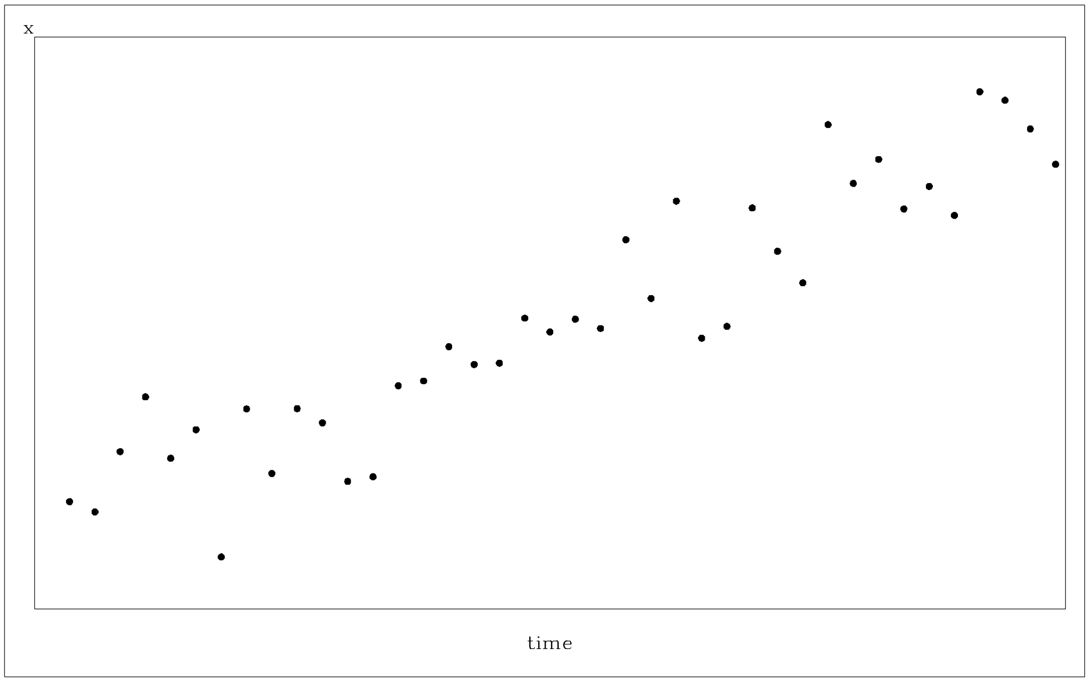


Figure 17.9. Time Series with Linear Trend

A linear model is appropriate for this data. For the linear model, assume the x_t values are generated according to the equation

$$x_t = b_0 + b_1 t + \epsilon_t$$

The linear model has two parameters. The predicted values for the future are the points on the estimated line. The extension of the polynomial model to three parameters is the quadratic (which forms a parabola). This allows for a constantly changing slope, where the x_t values are generated according to the equation

$$x_t = b_0 + b_1 t + b_2 t^2 + \epsilon_t$$

PROC FORECAST can fit three types of time trend models: constant, linear, and quadratic. For other kinds of trend models, other SAS procedures can be used.

Exponential smoothing fits a time trend model using a smoothing scheme in which the weights decline geometrically as you go backward in time. The forecasts from exponential smoothing are a time trend, but the trend is based mostly on the recent observations instead of on all the observations equally. How well exponential smoothing works as a forecasting method depends on choosing a good smoothing weight for the series.

To specify the exponential smoothing method, use the METHOD=EXPO option. Single exponential smoothing produces forecasts with a constant trend (that is, no trend). Double exponential smoothing produces forecasts with a linear trend, and triple exponential smoothing produces a quadratic trend. Use the TREND= option with the METHOD=EXPO option to select single, double, or triple exponential smoothing.

The time trend model can be modified to account for regular seasonal fluctuations of the series about the trend. To capture seasonality, the trend model includes a seasonal parameter for each season. Seasonal models can be additive or multiplicative.

$$x_t = b_0 + b_1 t + s(t) + \epsilon_t \qquad \text{(Additive)}$$

$$x_t = (b_0 + b_1 t)s(t) + \epsilon_t \qquad \text{(Multiplicative)}$$

where *s(t)* is the seasonal parameter for the season corresponding to time *t*.

The Winters method is similar to exponential smoothing, but includes seasonal factors. The Winters method can use either additive or multiplicative seasonal factors. Like exponential smoothing, good results with the Winters method depend on choosing good smoothing weights for the series to be forecast.

To specify the multiplicative or additive versions of the Winters method, use the METHOD=WINTERS or METHOD=ADDWINTERS options, respectively. To specify seasonal factors to include in the model, use the SEASONS= option.

Many observed time series do not behave like constant, linear, or quadratic time trends. However, you can partially compensate for the inadequacies of the trend models by fitting time series models to the departures from the time trend, as described in the following sections.

Time Series Methods

Time series models assume the future value of a variable to be a linear function of past values. If the model is a function of past values for a finite number of periods, it is an *autoregressive model* and is written as follows:

$$x_t = a_0 + a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_p x_{t-p} + \epsilon_t$$

The coefficients a_i are *autoregressive parameters*. One of the simplest cases of this model is the random walk, where the series dances around in purely random jumps. This is illustrated in Figure 17.10.

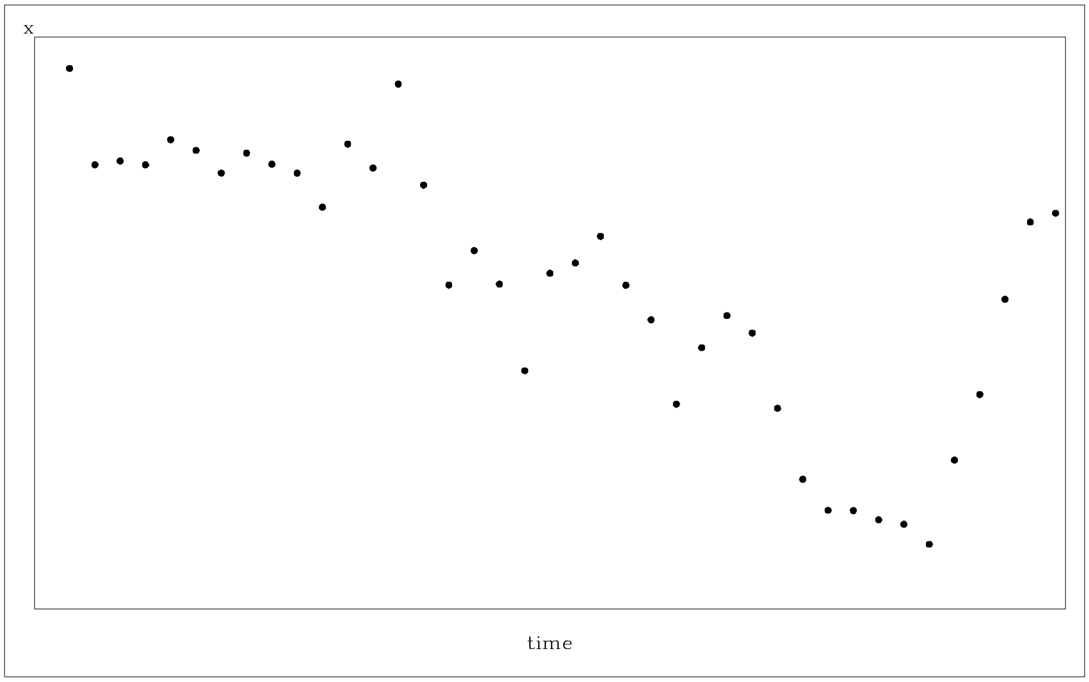


Figure 17.10. Random Walk Series

The x_t values are generated by the equation

$$x_t = x_{t-1} + \epsilon_t$$

In this type of model, the best forecast of a future value is the present value. However, with other autoregressive models, the best forecast is a weighted sum of recent values. Pure autoregressive forecasts always damp down to a constant (assuming the process is stationary).

Autoregressive time series models can also be used to predict seasonal fluctuations.

Combining Time Trend with Autoregressive Models

Trend models are suitable for capturing long-term behavior, whereas autoregressive models are more appropriate for capturing short-term fluctuations. One approach to forecasting is to combine a deterministic time trend model with an autoregressive model.

The *stepwise autoregressive method* (STEPAR method) combines a time-trend regression with an autoregressive model for departures from trend. The combined time-trend and autoregressive model is written as follows:

$$x_t = b_0 + b_1 t + b_2 t^2 + u_t$$

$$u_t = a_1 u_{t-1} + a_2 u_{t-2} + \ldots + a_p u_{t-p} + \epsilon_t$$

The autoregressive parameters included in the model for each series are selected by a stepwise regression procedure, so that autoregressive parameters are only included at those lags at which they are statistically significant.

The stepwise autoregressive method is fully automatic and, unlike the exponential smoothing and Winters methods, does not depend on choosing smoothing weights. However, the STEPAR method assumes that the long-term trend is stable; that is, the time trend regression is fit to the whole series with equal weights for the observations.

The stepwise autoregressive model is used when you specify the METHOD=STEPAR option or do not specify any METHOD= option. To select a constant, linear, or quadratic trend for the time-trend part of the model, use the TREND= option.

Syntax

The following statements are used with PROC FORECAST:

```
PROC FORECAST options;
   BY variables;
   ID variables;
   VAR variables;
```

Functional Summary

The statements and options controlling the FORECAST procedure are summarized in the following table:

Description	Statement	Option
Statements		
specify BY-group processing	BY	
identify observations	ID	
specify the variables to forecast	VAR	
Input Data Set Options		
specify the input SAS data set	PROC FORECAST	DATA=
specify frequency of the input time series	PROC FORECAST	INTERVAL=
specify increment between observations	PROC FORECAST	INTPER=
specify seasonality	PROC FORECAST	SEASONS=
specify number of periods in a season	PROC FORECAST	SINTPER=
treat zeros at beginning of series as missing	PROC FORECAST	ZEROMISS
Output Data Set Options		
specify the number of periods ahead to forecast	PROC FORECAST	LEAD=

Description	**Statement**	**Option**
name output data set containing the forecasts	PROC FORECAST	OUT=
write actual values to the OUT= data set	PROC FORECAST	OUTACTUAL
write confidence limits to the OUT= data set	PROC FORECAST	OUTLIMIT
write residuals to the OUT= data set	PROC FORECAST	OUTRESID
write standard errors of the forecasts to the OUT= data set	PROC FORECAST	OUTSTD
write one-step-ahead predicted values to the OUT= data set	PROC FORECAST	OUT1STEP
write predicted, actual, and confidence limit values to the OUT= data set	PROC FORECAST	OUTFULL
write all available results to the OUT= data set	PROC FORECAST	OUTALL
specify significance level for confidence limits	PROC FORECAST	ALPHA=
control the alignment of SAS Date values	PROC FORECAST	ALIGN=
Parameters and Statistics Output Data Set Options		
write parameter estimates and goodness-of-fit statistics to an output data set	PROC FORECAST	OUTEST=
write additional statistics to OUTEST= data set	PROC FORECAST	OUTESTALL
write Theil statistics to OUTEST= data set	PROC FORECAST	OUTESTTHEIL
write forecast accuracy statistics to OUTEST= data set	PROC FORECAST	OUTFITSTATS
Forecasting Method Options		
specify the forecasting method	PROC FORECAST	METHOD=
specify degree of the time trend model	PROC FORECAST	TREND=
specify smoothing weights	PROC FORECAST	WEIGHT=
specify order of the autoregressive model	PROC FORECAST	AR=
specify significance level for adding AR lags	PROC FORECAST	SLENTRY=
specify significance level for keeping AR lags	PROC FORECAST	SLSTAY=
start forecasting before the end of data	PROC FORECAST	START=
specify criterion for judging singularity	PROC FORECAST	SINGULAR=
Initializing Smoothed Values		
specify number of beginning values to use in calculating starting values	PROC FORECAST	NSTART=
specify number of beginning values to use in calculating initial seasonal parameters	PROC FORECAST	NSSTART=
specify starting values for constant term	PROC FORECAST	ASTART=
specify starting values for linear trend	PROC FORECAST	BSTART=
specify starting values for the quadratic trend	PROC FORECAST	CSTART=

PROC FORECAST Statement

PROC FORECAST *options;*

The following options can be specified in the PROC FORECAST statement:

ALIGN= *option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

ALPHA= *value*
specifies the significance level to use in computing the confidence limits of the forecast. The value of the ALPHA= option must be between .01 and .99. You should use only two digits for the ALPHA= option because PROC FORECAST rounds the value to the nearest percent (ALPHA=.101 is the same as ALPHA=.10). The default is ALPHA=.05, which produces 95% confidence limits.

AR= *n*
NLAGS= *n*
specifies the maximum order of the autoregressive model. The AR= option is only valid for METHOD=STEPAR. The default value of *n* depends on the INTERVAL= option and on the number of observations in the DATA= data set. See "STEPAR Method" later in this chapter for details.

ASTART= *value*
ASTART= *(value ...)*
specifies starting values for the constant term for the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR. See "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" later in this chapter for details.

BSTART= *value*
BSTART= *(value ...)*
specifies starting values for the linear trend for the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR or TREND=1. See "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" later in this chapter for details.

CSTART= *value*
CSTART= *(value ...)*
specifies starting values for the quadratic trend for the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR or TREND=1 or 2. See "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" later in this chapter for details.

DATA= *SAS-data-set*
names the SAS data set containing the input time series for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

INTERVAL= *interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. See Chapter 3, “Date Intervals, Formats, and Functions,” for more details on the intervals available.

INTPER= *n*

when the INTERVAL= option is not used, INTPER= specifies an increment (other than 1) to use in generating the values of the ID variable for the forecast observations in the output data set.

LEAD= *n*

specifies the number of periods ahead to forecast. The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the end of a particular series. Thus, if a series has missing values at the end, the actual number of forecasts computed for that series will be greater than the LEAD= value.

MAXERRORS= *n*

limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERRORS=50.

This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

METHOD= *method-name*

specifies the method to use to model the series and generate the forecasts.

METHOD=STEPAR specifies the stepwise autoregressive method.

METHOD=EXPO specifies the exponential smoothing method.

METHOD=WINTERS specifies the Holt-Winters exponentially smoothed trend-seasonal method.

METHOD=ADDWINTERS specifies the additive seasonal factors variant of the Winters method.

For more information, see the section "Forecasting Methods" later in this chapter. The default is METHOD=STEPAR.

NSTART= *n*
NSTART= MAX

specifies the number of beginning values of the series to use in calculating starting values for the trend parameters in the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR.

For METHOD=EXPO, *n* beginning values of the series are used in forming the exponentially smoothed values S1, S2, and S3, where *n* is the value of the NSTART= option. The parameters are initialized by fitting a time trend regression to the first *n* nonmissing values of the series.

For METHOD=WINTERS or METHOD=ADDWINTERS, *n* beginning complete seasonal cycles are used to compute starting values for the trend parameters. For example, for monthly data the seasonal cycle is one year, and NSTART=2 specifies that the first 24 observations at the beginning of each series are used for the time trend regression used to calculate starting values.

When NSTART=MAX is specified, all the observations are used. The default for METHOD=EXPO is NSTART=8; the default for METHOD=WINTERS or METHOD=ADDWINTERS is NSTART=2. See "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" later in this chapter for details.

NSSTART= *n*
NSSTART= MAX

specifies the number of beginning values of the series to use in calculating starting values for seasonal parameters for METHOD=WINTERS or METHOD=ADDWINTERS. The seasonal parameters are initialized by averaging over the first *n* values of the series for each season, where *n* is the value of the NSSTART= option. When NSSTART=MAX is specified, all the observations are used.

If NSTART= is specified, but NSSTART= is not, NSSTART= defaults to the value specified for NSTART=. If neither NSTART= nor NSSTART= is specified, then the default is NSSTART=2. This option is ignored if METHOD=STEPAR or METHOD=EXPO. See "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" later in this chapter for details.

OUT= *SAS-data-set*

names the output data set to contain the forecasts. If the OUT= option is not specified, the data set is named using the DATA*n* convention. See "OUT= Data Set" later in this chapter for details.

OUTACTUAL

writes the actual values to the OUT= data set.

OUTALL

provides all the output control options (OUTLIMIT, OUT1STEP, OUTACTUAL, OUTRESID, and OUTSTD).

OUTEST= *SAS-data-set*

names an output data set to contain the parameter estimates and goodness-of-fit statistics. When the OUTEST= option is not specified, the parameters and goodness-of-fit statistics are not stored. See "OUTEST= Data Set" later in this chapter for details.

OUTESTALL

writes additional statistics to the OUTEST= data set. This option is the same as specifying both OUTESTTHEIL and OUTFITSTATS.

OUTESTTHEIL

writes Theil forecast accuracy statistics to the OUTEST= data set.

OUTFITSTATS

writes various R^2-type forecast accuracy statistics to the OUTEST= data set.

OUTFULL
provides OUTACTUAL, OUT1STEP, and OUTLIMIT output control options in addition to the forecast values.

OUTLIMIT
writes the forecast confidence limits to the OUT= data set.

OUTRESID
writes the residuals (when available) to the OUT= data set.

OUTSTD
writes the standard errors of the forecasts to the OUT= data set.

OUT1STEP
writes the one-step-ahead predicted values to the OUT= data set.

SEASONS= *interval*
SEASONS= (*interval1 [interval2 [interval3]]* **)**
SEASONS= *n*
SEASONS= (*n1 [n2 [n3]]* **)**
specifies the seasonality for seasonal models. The *interval* can be QTR, MONTH, DAY, or HOUR, or multiples of these (QTR2, MONTH2, MONTH3, MONTH4, MONTH6, HOUR2, HOUR3, HOUR4, HOUR6, HOUR8, HOUR12).

Alternatively, seasonality can be specified by giving the length of the seasonal cycles. For example, SEASONS=3 means that every group of three observations forms a seasonal cycle. The SEASONS= option is valid only for METHOD=WINTERS or METHOD=ADDWINTERS. See "Specifying Seasonality" later in this chapter for details.

SINGULAR= *value*
gives the criterion for judging singularity. The default depends on the precision of the computer that you run SAS programs on.

SINTPER= *m*
SINTPER= (*m1 [m2 [m3]]* **)**
specifies the number of periods to combine in forming a season. For example, SEASONS=3 SINTPER=2 specifies that each group of two observations forms a season and that the seasonal cycle repeats every six observations. The SINTPER= option is valid only when the SEASONS= option is used. See "Specifying Seasonality" later in this chapter for details.

SLENTRY= *value*
controls the significance levels for entry of autoregressive parameters in the STEPAR method. The value of the SLENTRY= option must be between 0 and 1. The default is SLENTRY=0.2. See "STEPAR Method" later in this chapter for details.

SLSTAY= *value*
controls the significance levels for removal of autoregressive parameters in the STEPAR method. The value of the SLSTAY= option must be between 0 and 1. The default is SLSTAY=0.05. See "STEPAR Method" later in this chapter for details.

START= *n*

uses the first *n* observations to fit the model and begins forecasting with the *n*+1 observation.

TREND= *n*

specifies the degree of the time trend model. The value of the TREND= option must be 1, 2, or 3. TREND=1 selects the constant trend model; TREND=2 selects the linear trend model; and TREND=3 selects the quadratic trend model. The default is TREND=2, except for METHOD=EXPO, for which the default is TREND=3.

WEIGHT= *w*

WEIGHT= (*w1 [w2 [w3]]* **)**

specifies the smoothing weights for the EXPO, WINTERS, and ADDWINTERS methods. For the EXPO method, only one weight can be specified. For the WINTERS or ADDWINTERS method, *w1* gives the weight for updating the constant component, *w2* gives the weight for updating the linear and quadratic trend components, and *w3* gives the weight for updating the seasonal component. The *w2* and *w3* values are optional. Each value in the WEIGHT= option must be between 0 and 1. For default values, see "EXPO Method" and "WINTERS Method" later in this chapter.

ZEROMISS

treats zeros at the beginning of a series as missing values. For example, a product may be introduced at a date after the date of the first observation in the data set, and the sales variable for the product may be recorded as zero for the observations prior to the introduction date. The ZEROMISS option says to treat these initial zeros as missing values.

BY Statement

BY *variables;*

A BY statement can be used with PROC FORECAST to obtain separate analyses on observations in groups defined by the BY variables.

ID Statement

ID *variables;*

The first variable listed in the ID statement identifies observations in the input and output data sets. Usually, the first ID variable is a SAS date or datetime variable. Its values are interpreted and extrapolated according to the values of the INTERVAL= option. See "Data Periodicity and Time Intervals" later in this chapter for details.

If more than one ID variable is specified in the ID statement, only the first is used to identify the observations; the rest are just copied to the OUT= data set and will have missing values for forecast observations.

VAR Statement

VAR *variables;*

The VAR statement specifies the variables in the input data set that you want to forecast. If no VAR statement is specified, the procedure forecasts all numeric variables except the ID and BY variables.

Details

Missing Values

The treatment of missing values varies by method. For METHOD=STEPAR, missing values are tolerated in the series; the autocorrelations are estimated from the available data and tapered, if necessary. For the EXPO, WINTERS, and ADDWINTERS methods, missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are applied to the smoothing equations. For the WINTERS method, negative or zero values are treated as missing.

Data Periodicity and Time Intervals

The INTERVAL= option is used to establish the frequency of the time series. For example, INTERVAL=MONTH specifies that each observation in the input data set represents one month. If INTERVAL=MONTH2, each observation represents two months. Thus, there is a two-month time interval between each pair of successive observations, and the data frequency is bimonthly.

See Chapter 3, “Date Intervals, Formats, and Functions,” for details on the interval values supported.

The INTERVAL= option is used together with the ID statement to fully describe the observations that make up the time series. The first variable specified in the ID statement is used to identify the observations. Usually, SAS date or datetime values are used for this variable. PROC FORECAST uses the ID variable in the following ways:

- to validate the data periodicity. When the INTERVAL= option is specified, the ID variable is used to check the data and verify that successive observations have valid ID values corresponding to successive time intervals. When the INTERVAL= option is not used, PROC FORECAST verifies that the ID values are nonmissing and in ascending order. A warning message is printed when an invalid ID value is found in the input data set.
- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for January 1970 is followed by an observation for April 1970, there is a gap in the input data, with two observations omitted. When a gap in the input data is found, a warning message is printed, and PROC FORECAST processes missing values for each omitted input observation.

- to label the forecast observations in the output data set. The values of the ID variable for the forecast observations after the end of the input data set are extrapolated according to the frequency specifications of the INTERVAL= option. If the INTERVAL= option is not specified, the ID variable is extrapolated by incrementing the ID variable value for the last observation in the input data set by the INTPER= value, if specified, or by one.

The ALIGN= option controls the alignment of SAS dates. See Chapter 3, "Date Intervals, Formats, and Functions," for more information.

Forecasting Methods

This section explains the forecasting methods used by PROC FORECAST.

STEPAR Method

In the STEPAR method, PROC FORECAST first fits a time trend model to the series and takes the difference between each value and the estimated trend. (This process is called *detrending*.) Then, the remaining variation is fit using an autoregressive model.

The STEPAR method fits the autoregressive process to the residuals of the trend model using a backwards-stepping method to select parameters. Since the trend and autoregressive parameters are fit in sequence rather than simultaneously, the parameter estimates are not optimal in a statistical sense; however, the estimates are usually close to optimal, and the method is computationally inexpensive.

The STEPAR Algorithm

The STEPAR method consists of the following computational steps:

1. Fit the trend model as specified by the TREND= option using ordinary least-squares regression. This step detrends the data. The default trend model for the STEPAR method is TREND=2, a linear trend model.
2. Take the residuals from step 1 and compute the autocovariances to the number of lags specified by the NLAGS= option.
3. Regress the current values against the lags, using the autocovariances from step 2 in a Yule-Walker framework. Do not bring in any autoregressive parameter that is not significant at the level specified by the SLENTRY= option. (The default is SLENTRY=0.20.) Do not bring in any autoregressive parameter which results in a nonpositive-definite Toeplitz matrix.
4. Find the autoregressive parameter that is least significant. If the significance level is greater than the SLSTAY= value, remove the parameter from the model. (The default is SLSTAY=0.05.) Continue this process until only significant autoregressive parameters remain. If the OUTEST= option is specified, write the estimates to the OUTEST= data set.
5. Generate the forecasts using the estimated model and output to the OUT= data set. Form the confidence limits by combining the trend variances with the autoregressive variances.

Missing values are tolerated in the series; the autocorrelations are estimated from the available data and tapered if necessary.

This method requires at least three passes through the data: two passes to fit the model and a third pass to initialize the autoregressive process and write to the output data set.

Default Value of the NLAGS= Option

If the NLAGS= option is not specified, the default value of the NLAGS= option is chosen based on the data frequency specified by the INTERVAL= option and on the number of observations in the input data set, if this can be determined in advance. (PROC FORECAST cannot determine the number of input observations before reading the data when a BY statement or a WHERE statement is used or if the data are from a tape format SAS data set or external database. The NLAGS= value must be fixed before the data are processed.)

If the INTERVAL= option is specified, the default NLAGS= value includes lags for up to three years plus one, subject to the maximum of 13 lags or one third of the number of observations in your data set, whichever is less. If the number of observations in the input data set cannot be determined, the maximum NLAGS= default value is 13. If the INTERVAL= option is not specified, the default is NLAGS=13 or one-third the number of input observations, whichever is less.

If the Toeplitz matrix formed by the autocovariance matrix at a given step is not positive definite, the maximal number of autoregressive lags is reduced.

For example, for INTERVAL=QTR, the default is NLAGS=13 (that is, $4{\times}3+1$) provided that there are at least 39 observations. The NLAGS= option default is always at least 3.

EXPO Method

Exponential smoothing is used when the METHOD=EXPO option is specified. The term *exponential smoothing* is derived from the computational scheme developed by Brown and others (Brown and Meyers 1961; Brown 1962). Estimates are computed with updating formulas that are developed across time series in a manner similar to smoothing.

The EXPO method fits a trend model such that the most recent data are weighted more heavily than data in the early part of the series. The weight of an observation is a geometric (exponential) function of the number of periods that the observation extends into the past relative to the current period. The weight function is

$$w_\tau = \omega(1-\omega)^{t-\tau}$$

where τ is the observation number of the past observation, *t* is the current observation number, and ω is the weighting constant specified with the WEIGHT= option.

You specify the model with the TREND= option as follows:

- TREND=1 specifies single exponential smoothing (a constant model)

- TREND=2 specifies double exponential smoothing (a linear trend model)
- TREND=3 specifies triple exponential smoothing (a quadratic trend model)

Updating Equations

The single exponential smoothing operation is expressed by the formula

$$S_t = \omega x_t + (1 - \omega)S_{t-1}$$

where S_t is the smoothed value at the current period, *t* is the time index of the current period, and x_t is the current actual value of the series. The smoothed value S_t is the forecast of x_{t+1} and is calculated as the smoothing constant ω times the value of the series, x_t, in the current period plus $(1 - \omega)$ times the previous smoothed value S_{t-1}, which is the forecast of x_t computed at time $t - 1$.

Double and triple exponential smoothing are derived by applying exponential smoothing to the smoothed series, obtaining smoothed values as follows:

$$S_t^{[2]} = \omega S_t + (1 - \omega)S_{t-1}^{[2]}$$

$$S_t^{[3]} = \omega S_t^{[2]} + (1 - \omega)S_{t-1}^{[3]}$$

Missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted value is then applied to the smoothing equations.

The polynomial time trend parameters CONSTANT, LINEAR, and QUAD in the OUTEST= data set are computed from S_T, $S_T^{[2]}$, and $S_T^{[3]}$, the final smoothed values at observation *T*, the last observation used to fit the model. In the OUTEST= data set, the values of S_T, $S_T^{[2]}$, and $S_T^{[3]}$ are identified by _TYPE_=S1, _TYPE_=S2, and _TYPE_=S3, respectively.

Smoothing Weights

Exponential smoothing forecasts are forecasts for an integrated moving-average process; however, the weighting parameter is specified by the user rather than estimated from the data. Experience has shown that good values for the WEIGHT= option are between 0.05 and 0.3. As a general rule, smaller smoothing weights are appropriate for series with a slowly changing trend, while larger weights are appropriate for volatile series with a rapidly changing trend. If unspecified, the weight defaults to $(1 - .8^{1/trend})$, where *trend* is the value of the TREND= option. This produces defaults of WEIGHT=0.2 for TREND=1, WEIGHT=0.10557 for TREND=2, and WEIGHT=0.07168 for TREND=3.

Confidence Limits

The confidence limits for exponential smoothing forecasts are calculated as they would be for an exponentially weighted time-trend regression, using the simplifying assumption of an infinite number of observations. The variance estimate is computed using the mean square of the unweighted one-step-ahead forecast residuals.

More detailed descriptions of the forecast computations can be found in Montgomery and Johnson (1976) and Brown (1962).

Exponential Smoothing as an ARIMA Model

The traditional description of exponential smoothing given in the preceding section is standard in most books on forecasting, and so this traditional version is employed by PROC FORECAST.

However, the standard exponential smoothing model is, in fact, a special case of an ARIMA model (McKenzie 1984). Single exponential smoothing corresponds to an ARIMA(0,1,1) model; double exponential smoothing corresponds to an ARIMA(0,2,2) model; and triple exponential smoothing corresponds to an ARIMA(0,3,3) model.

The traditional exponential smoothing calculations can be viewed as a simple and computationally inexpensive method of forecasting the equivalent ARIMA model. The exponential smoothing technique was developed in the 1960s before computers were widely available and before ARIMA modeling methods were developed.

If you use exponential smoothing as a forecasting method, you might consider using the ARIMA procedure to forecast the equivalent ARIMA model as an alternative to the traditional version of exponential smoothing used by PROC FORECAST. The advantages of the ARIMA form are:

- The optimal smoothing weight is automatically computed as the estimate of the moving average parameter of the ARIMA model.
- For double exponential smoothing, the optimal pair of two smoothing weights are computed. For triple exponential smoothing, the optimal three smoothing weights are computed by the ARIMA method. Most implementations of the traditional exponential smoothing method (including PROC FORECAST) use the same smoothing weight for each stage of smoothing.
- The problem of setting the starting smoothed value is automatically handled by the ARIMA method. This is done in a statistically optimal way when the maximum likelihood method is used.
- The statistical estimates of the forecast confidence limits have a sounder theoretical basis.

See Chapter 11, "The ARIMA Procedure," for information on forecasting with ARIMA models.

The Time Series Forecasting System provides for exponential smoothing models and allows you to either specify or optimize the smoothing weights. See Chapter 34, "Getting Started with Time Series Forecasting," for details.

WINTERS Method

The WINTERS method uses updating equations similar to exponential smoothing to fit parameters for the model

$$x_t = (a + bt)s(t) + \epsilon_t$$

where *a* and *b* are the trend parameters, and the function *s*(*t*) selects the seasonal parameter for the season corresponding to time *t*.

The WINTERS method assumes that the series values are positive. If negative or zero values are found in the series, a warning is printed and the values are treated as missing.

The preceding standard WINTERS model uses a linear trend. However, PROC FORECAST can also fit a version of the WINTERS method that uses a quadratic trend. When TREND=3 is specified for METHOD=WINTERS, PROC FORECAST fits the following model:

$$x_t = (a + bt + ct^2)s(t) + \epsilon_t$$

The quadratic trend version of the Winters method is often unstable, and its use is not recommended.

When TREND=1 is specified, the following constant trend version is fit:

$$x_t = as(t) + \epsilon_t$$

The default for the WINTERS method is TREND=2, which produces the standard linear trend model.

Seasonal Factors

The notation *s*(*t*) represents the selection of the seasonal factor used for different time periods. For example, if INTERVAL=DAY and SEASONS=MONTH, there are 12 seasonal factors, one for each month in the year, and the time index *t* is measured in days. For any observation, *t* is determined by the ID variable and *s*(*t*) selects the seasonal factor for the month that *t* falls in. For example, if *t* is 9 February 1993 then *s*(*t*) is the seasonal parameter for February.

When there are multiple seasons specified, *s*(*t*) is the product of the parameters for the seasons. For example, if SEASONS=(MONTH DAY), then *s*(*t*) is the product of the seasonal parameter for the month corresponding to the period *t*, and the seasonal parameter for the day of the week corresponding to period *t*. When the SEASONS= option is not specified, the seasonal factors *s*(*t*) are not included in the model. See the section "Specifying Seasonality" later in this chapter for more information on specifying multiple seasonal factors.

Updating Equations

This section shows the updating equations for the Winters method. In the following formula, x_t is the actual value of the series at time *t*; a_t is the smoothed value of the series at time *t*; b_t is the smoothed trend at time *t*; c_t is the smoothed quadratic trend at time *t*; $s_{t-1}(t)$ selects the old value of the seasonal factor corresponding to time *t* before the seasonal factors are updated.

The estimates of the constant, linear, and quadratic trend parameters are updated using the following equations:

For TREND=3,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1-\omega_1)(a_{t-1} + b_{t-1} + c_{t-1})$$

$$b_t = \omega_2(a_t - a_{t-1} + c_{t-1}) + (1-\omega_2)(b_{t-1} + 2c_{t-1})$$

$$c_t = \omega_2 \frac{1}{2}(b_t - b_{t-1}) + (1-\omega_2)c_{t-1}$$

For TREND=2,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1-\omega_1)(a_{t-1} + b_{t-1})$$

$$b_t = \omega_2(a_t - a_{t-1}) + (1-\omega_2)b_{t-1}$$

For TREND=1,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1-\omega_1)a_{t-1}$$

In this updating system, the trend polynomial is always centered at the current period so that the intercept parameter of the trend polynomial for predicted values at times after *t* is always the updated intercept parameter a_t. The predicted value for τ periods ahead is

$$x_{t+\tau} = (a_t + b_t\tau)s_t(t+\tau)$$

The seasonal parameters are updated when the season changes in the data, using the mean of the ratios of the actual to the predicted values for the season. For example, if SEASONS=MONTH and INTERVAL=DAY, then when the observation for the first of February is encountered, the seasonal parameter for January is updated using the formula

$$s_t(t-1) = \omega_3 \frac{1}{31} \sum_{i=t-31}^{t-1} \frac{x_i}{a_i} + (1-\omega_3)s_{t-1}(t-1)$$

where *t* is February 1 of the current year, $s_t(t-1)$ is the seasonal parameter for January updated with the data available at time *t*, and $s_{t-1}(t-1)$ is the seasonal parameter for January of the previous year.

When multiple seasons are used, $s_t(t)$ is a product of seasonal factors. For example, if SEASONS=(MONTH DAY) then $s_t(t)$ is the product of the seasonal factors for the month and for the day of the week: $s_t(t) = s_t^m(t)s_t^d(t)$.

The factor $s_t^m(t)$ is updated at the start of each month using a modification of the preceding formula that adjusts for the presence of the other seasonal by dividing the summands $\frac{x_i}{a_i}$ by the corresponding day of the week effect $s_i^d(i)$.

Similarly, the factor $s_t^d(t)$ is updated using the following formula:

$$s_t^d(t) = \omega_3 \frac{x_t}{a_t s_t^m(t)} + (1 - \omega_3) s_{t-1}^d(t)$$

where $s_{t-1}^d(t)$ is the seasonal factor for the same day of the previous week.

Missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted value is substituted for x_i and applied to the updating equations.

Normalization

The parameters are normalized so that the seasonal factors for each cycle have a mean of 1.0. This normalization is performed after each complete cycle and at the end of the data. Thus, if INTERVAL=MONTH and SEASONS=MONTH are specified, and a series begins with a July value, then the seasonal factors for the series are normalized at each observation for July and at the last observation in the data set. The normalization is performed by dividing each of the seasonal parameters, and multiplying each of the trend parameters, by the mean of the unnormalized seasonal parameters.

Smoothing Weights

The weight for updating the seasonal factors, ω_3, is given by the third value specified in the WEIGHT= option. If the WEIGHT= option is not used, then ω_3 defaults to 0.25; if the WEIGHT= option is used but does not specify a third value, then ω_3 defaults to ω_2. The weight for updating the linear and quadratic trend parameters, ω_2, is given by the second value specified in the WEIGHT= option; if the WEIGHT= option does not specify a second value, then ω_2 defaults to ω_1. The updating weight for the constant parameter, ω_1, is given by the first value specified in the WEIGHT= option. As a general rule, smaller smoothing weights are appropriate for series with a slowly changing trend, while larger weights are appropriate for volatile series with a rapidly changing trend.

If the WEIGHT= option is not used, then ω_1 defaults to $(1 - .8^{1/trend})$, where *trend* is the value of the TREND= option. This produces defaults of WEIGHT=0.2 for TREND=1, WEIGHT=0.10557 for TREND=2, and WEIGHT=0.07168 for TREND=3.

The Time Series Forecasting System provides for generating forecast models using Winters Method and allows you to specify or optimize the weights. See Chapter 34, "Getting Started with Time Series Forecasting," for details.

Confidence Limits

A method for calculating exact forecast confidence limits for the WINTERS method is not available. Therefore, the approach taken in PROC FORECAST is to assume that the true seasonal factors have small variability about a set of fixed seasonal factors and that the remaining variation of the series is small relative to the mean level of the series. The equations are written

$$s_t(t) = \mathrm{I(t)}(1 + \delta_{\mathrm{t}})$$

$$x_t = \mu \mathrm{I(t)}(1 + \gamma_{\mathrm{t}})$$

$$a_t = \xi(1 + \alpha_t)$$

where μ is the mean level and I(*t*) are the fixed seasonal factors. Assuming that α_t and δ_t are small, the forecast equations can be linearized and only first-order terms in δ_t and α_t kept. In terms of forecasts for γ_t, this linearized system is equivalent to a seasonal ARIMA model. Confidence limits for γ_t are based on this ARIMA model and converted into confidence limits for x_t using $s_t(t)$ as estimates of I(*t*).

The exponential smoothing confidence limits are based on an approximation to a weighted regression model, whereas the preceding Winters confidence limits are based on an approximation to an ARIMA model. You can use METHOD=WINTERS without the SEASONS= option to do exponential smoothing and get confidence limits for the EXPO forecasts based on the ARIMA model approximation. These are generally more pessimistic than the weighted regression confidence limits produced by METHOD=EXPO.

ADDWINTERS Method

The ADDWINTERS method is like the WINTERS method except that the seasonal parameters are added to the trend instead of multiplied with the trend. The default TREND=2 model is as follows:

$$x_t = a + bt + s(t) + \epsilon_t$$

The WINTERS method for updating equation and confidence limits calculations described in the preceding section are modified accordingly for the additive version.

Holt Two-Parameter Exponential Smoothing

If the seasonal factors are omitted (that is, if the SEASONS= option is not specified), the WINTERS (and ADDWINTERS) method reduces to the Holt two-parameter version of exponential smoothing. Thus, the WINTERS method is often referred to as the Holt-Winters method.

Double exponential smoothing is a special case of the Holt two-parameter smoother. The double exponential smoothing results can be duplicated with METHOD=WINTERS by omitting the SEASONS= option and appropriately setting the WEIGHT= option. Letting $\alpha = \omega(2 - \omega)$ and $\beta = \omega/(2 - \omega)$, the following statements produce the same forecasts:

```
   proc forecast method=expo trend=2 weight=ω ...  ;
   proc forecast method=winters trend=2
                 weight=(α,β) ...  ;
```

Although the forecasts are the same, the confidence limits are computed differently.

Choice of Weights for EXPO, WINTERS, and ADDWINTERS Methods

For the EXPO, WINTERS, and ADDWINTERS methods, properly chosen smoothing weights are of critical importance in generating reasonable results. There are several factors to consider in choosing the weights.

The noisier the data, the lower should be the weight given to the most recent observation. Another factor to consider is how quickly the mean of the time series is changing. If the mean of the series is changing rapidly, relatively more weight should be given to the most recent observation. The more stable the series over time, the lower should be the weight given to the most recent observation.

Note that the smoothing weights should be set separately for each series; weights that produce good results for one series may be poor for another series. Since PROC FORECAST does not have a feature to use different weights for different series, when forecasting multiple series with the EXPO, WINTERS, or ADDWINTERS method it may be desirable to use different PROC FORECAST steps with different WEIGHT= options.

For the Winters method, many combinations of weight values may produce unstable *noninvertible* models, even though all three weights are between 0 and 1. When the model is noninvertible, the forecasts depend strongly on values in the distant past, and predictions are determined largely by the starting values. Unstable models usually produce poor forecasts. The Winters model may be unstable even if the weights are optimally chosen to minimize the in-sample MSE. Refer to Archibald (1990) for a detailed discussion of the unstable region of the parameter space of the Winters model.

Optimal weights and forecasts for exponential smoothing models can be computed using the ARIMA procedure. For more information, see "Exponential Smoothing as an ARIMA Model" earlier in this chapter.

The ARIMA procedure can also be used to compute optimal weights and forecasts for seasonal ARIMA models similar to the Winters type methods. In particular, an ARIMA(0,1,1)$\times$(0,1,1)S model may be a good alternative to the additive version of the Winters method. The ARIMA(0,1,1)$\times$(0,1,1)S model fit to the logarithms of the series may be a good alternative to the multiplicative Winters method. See Chapter 11, “The ARIMA Procedure,” for information on forecasting with ARIMA models.

The Time Series Forecasting System can be used to automatically select an appropriate smoothing method as well as to optimize the smoothing weights. See Chapter 34, “Getting Started with Time Series Forecasting,” for more information.

Starting Values for EXPO, WINTERS, and ADDWINTERS Methods

The exponential smoothing method requires starting values for the smoothed values S_0, $S_0^{[2]}$, and $S_0^{[3]}$. The Winters and additive Winters methods require starting values for the trend coefficients and seasonal factors.

By default, starting values for the trend parameters are computed by a time-trend regression over the first few observations for the series. Alternatively, you can specify the starting value for the trend parameters with the ASTART=, BSTART=, and CSTART= options.

The number of observations used in the time-trend regression for starting values depends on the NSTART= option. For METHOD=EXPO, NSTART= beginning values of the series are used, and the coefficients of the time-trend regression are then used to form the initial smoothed values S_0, $S_0^{[2]}$, and $S_0^{[3]}$.

For METHOD=WINTERS or METHOD=ADDWINTERS, *n* complete seasonal cycles are used to compute starting values for the trend parameter, where *n* is the value of the NSTART= option. For example, for monthly data the seasonal cycle is one year, so NSTART=2 specifies that the first 24 observations at the beginning of each series are used for the time trend regression used to calculate starting values.

The starting values for the seasonal factors for the WINTERS and ADDWINTERS methods are computed from seasonal averages over the first few complete seasonal cycles at the beginning of the series. The number of seasonal cycles averaged to compute starting seasonal factors is controlled by the NSSTART= option. For example, for monthly data with SEASONS=12 or SEASONS=MONTH, the first *n* January values are averaged to get the starting value for the January seasonal parameter, where *n* is the value of the NSSTART= option.

The $s_0(i)$ seasonal parameters are set to the ratio (for WINTERS) or difference (for ADDWINTERS) of the mean for the season to the overall mean for the observations used to compute seasonal starting values.

For example, if METHOD=WINTERS, INTERVAL=DAY, SEASON=(MONTH DAY), and NSTART=2 (the default), the initial seasonal parameter for January is the ratio of the mean value over days in the first two Januarys after the start of the series (that is, after the first nonmissing value), to the mean value for all days read for initialization of the seasonal factors. Likewise, the initial factor for Sundays is the ratio of the mean value for Sundays to the mean of all days read.

For the ASTART=, BSTART=, and CSTART= options, the values specified are associated with the variables in the VAR statement in the order in which the variables are listed (the first value with the first variable, the second value with the second variable, and so on). If there are fewer values than variables, default starting values are used for the later variables. If there are more values than variables, the extra values are ignored.

Specifying Seasonality

Seasonality of a time series is a regular fluctuation about a trend. This is called seasonality because the time of year is the most common source of periodic variation. For example, sales of home heating oil are regularly greater in winter than during other times of the year.

Seasonality can be caused by many things other than weather. In the United States, sales of nondurable goods are greater in December than in other months because of the Christmas shopping season. The term seasonality is also used for cyclical fluctuation at periods other than a year. Often, certain days of the week cause regular fluctuation in daily time series, such as increased spending on leisure activities during weekends.

Three kinds of seasonality are supported in PROC FORECAST: time-of-year, day-of-week, and time-of-day. The seasonal part of the model is specified using the SEASONS= option. The values for the SEASONS= option are listed in Table 17.1.

Table 17.1. The SEASONS= Option

SEASONS= Value	Cycle Length	Type of Seasonality
QTR	yearly	time of year
MONTH	yearly	time of year
DAY	weekly	day of week
HOUR	daily	time of day

The three kinds of seasonality can be combined. For example, SEASONS=(MONTH DAY HOUR) specifies that 24 hour-of-day seasons are nested within 7 day-of-week seasons, which in turn are nested within 12 month-of-year seasons. The different kinds of intervals can be listed in the SEASONS= option in any order. Thus, SEASONS=(HOUR DAY MONTH) is the same as SEASONS=(MONTH DAY HOUR). Note that the Winters method smoothing equations may be less stable when multiple seasonal factors are used.

Multiple period seasons can also be used. For example, SEASONS=QTR2 specifies two semiannual time-of-year seasons. The grouping of observations into multiple period seasons starts with the first interval in the seasonal cycle. Thus, MONTH2 seasons are January-February, March-April, and so on. (There is no provision for shifting seasonal intervals; thus, there is no way to specify December-January, February-March, April-May, and so on seasons.)

For multiple period seasons, the number of intervals combined to form the seasons must evenly divide and be less than the basic cycle length. For example, with SEASONS=MONTH*n*, the basic cycle length is 12, so MONTH2, MONTH3, MONTH4, and MONTH6 are valid SEASONS= values (since 2, 3, 4, and 6 evenly divide 12 and are less than 12), but MONTH5 and MONTH12 are not valid SEASONS= values.

The frequency of the seasons must not be greater than the frequency of the input data. For example, you cannot specify SEASONS=MONTH if INTERVAL=QTR or SEASONS=MONTH if INTERVAL=MONTH2. You also cannot specify two seasons of the same basic cycle. For example, SEASONS=(MONTH QTR) or SEASONS=(MONTH2 MONTH4) is not allowed.

Alternatively, the seasonality can be specified by giving the number of seasons in the SEASONS= option. SEASONS=*n* specifies that there are *n* seasons, with observations 1, *n*+1, 2*n*+1, and so on in the first season, observations 2, *n*+2, 2*n*+2, and so on in the second season, and so forth.

The options SEASONS=*n* and SINTPER=*m* cause PROC FORECAST to group the input observations into *n* seasons, with *m* observations to a season, which repeat every

nm observations. The options SEASONS=(n_1 n_2) and SINTPER=(m_1 m_2) produce n_1 seasons with m_1 observations to a season nested within n_2 seasons with $n_1 m_1 m_2$ observations to a season.

If the SINTPER=m option is used with the SEASONS= option, the SEASONS= interval is multiplied by the SINTPER= value. For example, specifying both SEASONS=(QTR HOUR) and SINTPER=(2 3) is the same as specifying SEASONS=(QTR2 HOUR3) and also the same as specifying SEASONS=(HOUR3 QTR2).

Data Requirements

You should have ample data for the series that you forecast using PROC FORECAST. However, the results may be poor unless you have a good deal more than the minimum amount of data the procedure allows. The minimum number of observations required for the different methods is as follows:

- If METHOD=STEPAR is used, the minimum number of nonmissing observations required for each series forecast is the TREND= option value plus the value of the NLAGS= option. For example, using NLAGS=13 and TREND=2, at least 15 nonmissing observations are needed.
- If METHOD=EXPO is used, the minimum is the TREND= option value.
- If METHOD=WINTERS or ADDWINTERS is used, the minimum number of observations is either the number of observations in a complete seasonal cycle or the TREND= option value, whichever is greater. (However, there should be data for several complete seasonal cycles, or the seasonal factor estimates may be poor.) For example, for the seasonal specifications SEASONS=MONTH, SEASONS=(QTR DAY), or SEASONS=(MONTH DAY HOUR), the longest cycle length is one year, so at least one year of data is required.

OUT= Data Set

The FORECAST procedure writes the forecast to the output data set named by the OUT= option. The OUT= data set contains the following variables:

- the BY variables
- _TYPE_, a character variable that identifies the type of observation
- _LEAD_, a numeric variable that indicates the number of steps ahead in the forecast. The value of _LEAD_ is 0 for the one-step-ahead forecasts before the start of the forecast period.
- the ID statement variables
- the VAR statement variables, which contain the result values as indicated by the _TYPE_ variable value for the observation

The FORECAST procedure processes each of the input variables listed in the VAR statement and writes several observations for each forecast period to the OUT= data set. The observations are identified by the value of the _TYPE_ variable. The options OUTACTUAL, OUTALL, OUTLIMIT, OUTRESID, OUT1STEP, OUTFULL, and OUTSTD control which types of observations are included in the OUT= data set.

The values of the variable _TYPE_ are as follows:

ACTUAL	The VAR statement variables contain actual values from the input data set. The OUTACTUAL option writes the actual values. By default, only the observations for the forecast period are output.
FORECAST	The VAR statement variables contain forecast values. The OUT1STEP option writes the one-step-ahead predicted values for the observations used to fit the model.
RESIDUAL	The VAR statement variables contain residuals. The residuals are computed by subtracting the forecast value from the actual value ($residual = actual - forecast$). The OUTRESID option writes observations for the residuals.
L*nn*	The VAR statement variables contain lower *nn*% confidence limits for the forecast values. The value of *nn* depends on the ALPHA= option; with the default ALPHA=0.05, the _TYPE_ value is L95 for the lower confidence limit observations. The OUTLIMIT option writes observations for the upper and lower confidence limits.
U*nn*	The VAR statement variables contain upper *nn*% confidence limits for the forecast values. The value of *nn* depends on the ALPHA= option; with the default ALPHA=0.05, the _TYPE_ value is U95 for the upper confidence limit observations. The OUTLIMIT option writes observations for the upper and lower confidence limits.
STD	The VAR statement variables contain standard errors of the forecast values. The OUTSTD option writes observations for the standard errors of the forecast.

If no output control options are specified, PROC FORECAST outputs only the forecast values for the forecast periods.

The _TYPE_ variable can be used to subset the OUT= data set. For example, the following data step splits the OUT= data set into two data sets, one containing the forecast series and the other containing the residual series. For example

```
proc forecast out=out outresid ...;
   ...
run;

data fore resid;
   set out;
   if _TYPE_='FORECAST' then output fore;
   if _TYPE_='RESIDUAL' then output resid;
run;
```

See Chapter 2, “Working with Time Series Data,” for more information on processing time series data sets in this format.

OUTEST= Data Set

The FORECAST procedure writes the parameter estimates and goodness-of-fit statistics to an output data set when the OUTEST= option is specified. The OUTEST= data set contains the following variables:

- the BY variables
- the first ID variable, which contains the value of the ID variable for the last observation in the input data set used to fit the model
- _TYPE_, a character variable that identifies the type of each observation
- the VAR statement variables, which contain statistics and parameter estimates for the input series. The values contained in the VAR statement variables depend on the _TYPE_ variable value for the observation.

The observations contained in the OUTEST= data set are identified by the _TYPE_ variable. The OUTEST= data set may contain observations with the following _TYPE_ values:

AR1–AR*n*	The observation contains estimates of the autoregressive parameters for the series. Two-digit lag numbers are used if the value of the NLAGS= option is 10 or more; in that case these _TYPE_ values are AR01–AR*n*. These observations are output for the STEPAR method only.
CONSTANT	The observation contains the estimate of the constant or intercept parameter for the time-trend model for the series. For the exponential smoothing and the Winters’ methods, the trend model is centered (that is, t=0) at the last observation used for the fit.
LINEAR	The observation contains the estimate of the linear or slope parameter for the time-trend model for the series. This observation is output only if you specify TREND=2 or TREND=3.
N	The observation contains the number of nonmissing observations used to fit the model for the series.
QUAD	The observation contains the estimate of the quadratic parameter for the time-trend model for the series. This observation is output only if you specify TREND=3.
SIGMA	The observation contains the estimate of the standard deviation of the error term for the series.
S1–S3	The observations contain exponentially smoothed values at the last observation. _TYPE_=S1 is the final smoothed value of the single exponential smooth. _TYPE_=S2 is the final smoothed value of the double exponential smooth. _TYPE_=S3 is the final smoothed

	value of the triple exponential smooth. These observations are output for METHOD=EXPO only.
S_*name*	The observation contains estimates of the seasonal parameters. For example, if SEASONS=MONTH, the OUTEST= data set will contain observations with _TYPE_=S_JAN, _TYPE_=S_FEB, _TYPE_=S_MAR, and so forth. For multiple-period seasons, the names of the first and last interval of the season are concatenated to form the season name. Thus, for SEASONS=MONTH4, the OUTEST= data set will contain observations with _TYPE_=S_JANAPR, _TYPE_=S_MAYAUG, and _TYPE_=S_SEPDEC. When the SEASONS= option specifies numbers, the seasonal factors are labeled _TYPE_=S_*i*_*j*. For example, SEASONS=(2 3) produces observations with _TYPE_ values of S_1_1, S_1_2, S_2_1, S_2_2, and S_2_3. The observation with _TYPE_=S_*i*_*j* contains the seasonal parameters for the *j*th season of the *i*th seasonal cycle. These observations are output only for METHOD=WINTERS or METHOD=ADDWINTERS.
WEIGHT	The observation contains the smoothing weight used for exponential smoothing. This is the value of the WEIGHT= option. This observation is output for METHOD=EXPO only.
WEIGHT1	
WEIGHT2	
WEIGHT3	The observations contain the weights used for smoothing the WINTERS or ADDWINTERS method parameters (specified by the WEIGHT= option). _TYPE_=WEIGHT1 is the weight used to smooth the CONSTANT parameter. _TYPE_=WEIGHT2 is the weight used to smooth the LINEAR and QUAD parameters. _TYPE_=WEIGHT3 is the weight used to smooth the seasonal parameters. These observations are output only for the WINTERS and ADDWINTERS methods.
NRESID	The observation contains the number of nonmissing residuals, *n*, used to compute the goodness-of-fit statistics. The residuals are obtained by subtracting the one-step-ahead predicted values from the observed values.
SST	The observation contains the total sum of squares for the series, corrected for the mean. $SST = \sum_{t=0}^{n} (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean.
SSE	The observation contains the sum of the squared residuals, uncorrected for the mean. $SSE = \sum_{t=0}^{n} (y_t - \hat{y}_t)^2$, where $\hat{y}$ is the one-step predicted value for the series.

MSE	The observation contains the mean squared error, calculated from one-step-ahead forecasts. $MSE = \frac{1}{n-k} SSE$, where *k* is the number of parameters in the model.
RMSE	The observation contains the root mean square error. $RMSE = \sqrt{MSE}$.
MAPE	The observation contains the mean absolute percent error. $MAPE = \frac{100}{n} \sum_{t=0}^{n} \lvert (y_t - \hat{y}_t)/y_t \rvert$.
MPE	The observation contains the mean percent error. $MPE = \frac{100}{n} \sum_{t=0}^{n} (y_t - \hat{y}_t)/y_t$.
MAE	The observation contains the mean absolute error. $MAE = \frac{1}{n} \sum_{t=0}^{n} \lvert y_t - \hat{y}_t \rvert$.
ME	The observation contains the mean error. $MAE = \frac{1}{n} \sum_{t=0}^{n} (y_t - \hat{y}_t)$.
MAXE	The observation contains the maximum error (the largest residual).
MINE	The observation contains the minimum error (the smallest residual).
MAXPE	The observation contains the maximum percent error.
MINPE	The observation contains the minimum percent error.
RSQUARE	The observation contains the R^2 statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares *SSE* may be larger than *SST* and the R^2 statistic will be negative.
ADJRSQ	The observation contains the adjusted R^2 statistic. $ADJRSQ = 1 - (\frac{n-1}{n-k})(1 - R^2)$.
ARSQ	The observation contains Amemiya's adjusted R^2 statistic. $ARSQ = 1 - (\frac{n+k}{n-k})(1 - R^2)$.
RW_RSQ	The observation contains the random walk R^2 statistic (Harvey's R_{D}^2 statistic using the random walk model for comparison). $RW\_RSQ = 1 - (\frac{n-1}{n}) SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$, and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$.
AIC	The observation contains Akaike's information criterion. $AIC = n\ln(SSE/n) + 2k$.
SBC	The observation contains Schwarz's Bayesian criterion. $SBC = n\ln(SSE/n) + k\ln(n)$.
APC	The observation contains Amemiya's prediction criterion. $APC = \frac{1}{n} SST(\frac{n+k}{n-k})(1 - R^2) = (\frac{n+k}{n-k})\frac{1}{n} SSE$.

CORR	The observation contains the correlation coefficient between the actual values and the one-step-ahead predicted values.
THEILU	The observation contains Theil's U statistic using original units. Refer to Maddala (1977, pp. 344-345), and Pindyck and Rubinfeld (1981, pp. 364-365) for more information on Theil statistics.
RTHEILU	The observation contains Theil's U statistic calculated using relative changes.
THEILUM	The observation contains the bias proportion of Theil's U statistic.
THEILUS	The observation contains the variance proportion of Theil's U statistic.
THEILUC	The observation contains the covariance proportion of Theil's U statistic.
THEILUR	The observation contains the regression proportion of Theil's U statistic.
THEILUD	The observation contains the disturbance proportion of Theil's U statistic.
RTHEILUM	The observation contains the bias proportion of Theil's U statistic, calculated using relative changes.
RTHEILUS	The observation contains the variance proportion of Theil's U statistic, calculated using relative changes.
RTHEILUC	The observation contains the covariance proportion of Theil's U statistic, calculated using relative changes.
RTHEILUR	The observation contains the regression proportion of Theil's U statistic, calculated using relative changes.
RTHEILUD	The observation contains the disturbance proportion of Theil's U statistic, calculated using relative changes.

Examples

Example 17.1. Forecasting Auto Sales

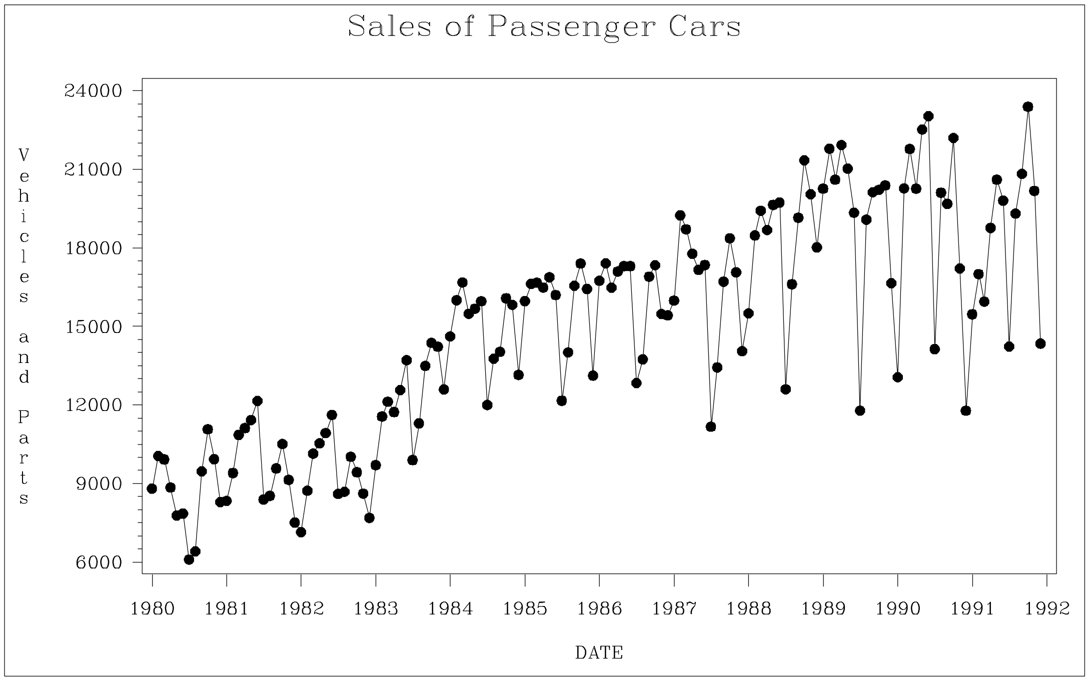
This example uses the Winters method to forecast the monthly U.S. sales of passenger cars series (VEHICLES) from the data set SASHELP.USECON. These data are taken from *Business Statistics*, published by the U.S. Bureau of Economic Analysis.

The following statements plot the series; the plot is shown in Output 17.1.1:

```
title1 "Sales of Passenger Cars";

symbol1 i=spline v=dot;
axis2 label=(a=-90 r=90 "Vehicles and Parts" )
      order=(6000 to 24000 by 3000) ;
proc gplot data=sashelp.usecon;
   plot vehicles * date = 1 /
```

```
          haxis= '1jan80'd to '1jan92'd by year
          vaxis=axis2;
     where date >= '1jan80'd;
     format date year4.;
  run;
```

Output 17.1.1. Monthly Passenger Car Sales



The following statements produce the forecast:

```
  proc forecast data=sashelp.usecon interval=month
                method=winters seasons=month lead=12
                out=out outfull outresid outest=est;
     id date;
     var vehicles;
     where date >= '1jan80'd;
  run;
```

The INTERVAL=MONTH option indicates that the data are monthly, and the ID DATE statement gives the dating variable. The METHOD=WINTERS specifies the Winters smoothing method. The LEAD=12 option forecasts 12 months ahead. The OUT=OUT option specifies the output data set, while the OUTFULL and OUTRESID options include in the OUT= data set the predicted and residual values for the historical period and the confidence limits for the forecast period. The OUTEST= option stores various statistics in an output data set. The WHERE statement is used to include only data from 1980 on.

The following statements print the OUT= data set:

```
title2 'The OUT= Data Set';
proc print data=out;
run;
```

The listing of the output data set produced by PROC PRINT is shown in part in Output 17.1.2.

Output 17.1.2. The OUT= Data Set Produced by PROC FORECAST

```
                        Sales of Passenger Cars
                           The OUT= Data Set

           Obs    DATE     _TYPE_      _LEAD_     VEHICLES

           421    SEP91    ACTUAL         0       20827.00
           422    SEP91    FORECAST       0       18266.20
           423    SEP91    RESIDUAL       0        2560.79
           424    OCT91    ACTUAL         0       23388.00
           425    OCT91    FORECAST       0       19913.88
           426    OCT91    RESIDUAL       0        3474.11
           427    NOV91    ACTUAL         0       20181.00
           428    NOV91    FORECAST       0       18294.58
           429    NOV91    RESIDUAL       0        1886.41
           430    DEC91    ACTUAL         0       14344.00
           431    DEC91    FORECAST       0       15172.36
           432    DEC91    RESIDUAL       0        -828.36
           433    JAN92    FORECAST       1       16555.17
           434    JAN92    L95            1       13514.26
           435    JAN92    U95            1       19596.08
           436    FEB92    FORECAST       2       19516.83
           437    FEB92    L95            2       15908.52
           438    FEB92    U95            2       23125.16
           439    MAR92    FORECAST       3       19607.89
           440    MAR92    L95            3       15954.55
           441    MAR92    U95            3       23261.22
```

The following statements print the OUTEST= data set:

```
title2 'The OUTEST= Data Set: WINTERS Method';
proc print data=est;
run;
```

The PROC PRINT listing of the OUTEST= data set is shown in Output 17.1.3.

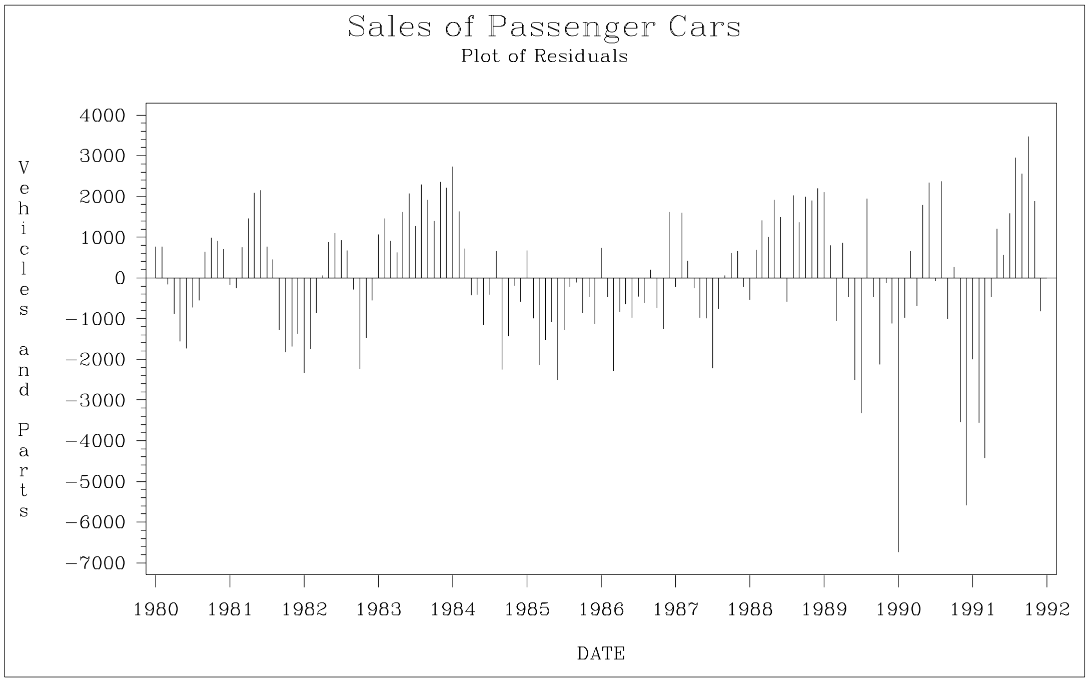
Output 17.1.3. The OUTEST= Data Set Produced by PROC FORECAST

```
                            Sales of Passenger Cars
                      The OUTEST= Data Set: WINTERS Method

                     Obs    _TYPE_      DATE      VEHICLES

                       1    N           DEC91          144
                       2    NRESID      DEC91          144
                       3    DF          DEC91          130
                       4    WEIGHT1     DEC91    0.1055728
                       5    WEIGHT2     DEC91    0.1055728
                       6    WEIGHT3     DEC91         0.25
                       7    SIGMA       DEC91     1741.481
                       8    CONSTANT    DEC91    18577.368
                       9    LINEAR      DEC91     4.804732
                      10    S_JAN       DEC91    0.8909173
                      11    S_FEB       DEC91    1.0500278
                      12    S_MAR       DEC91    1.0546539
                      13    S_APR       DEC91     1.074955
                      14    S_MAY       DEC91    1.1166121
                      15    S_JUN       DEC91    1.1012972
                      16    S_JUL       DEC91    0.7418297
                      17    S_AUG       DEC91    0.9633888
                      18    S_SEP       DEC91     1.051159
                      19    S_OCT       DEC91    1.1399126
                      20    S_NOV       DEC91    1.0132126
                      21    S_DEC       DEC91     0.802034
                      22    SST         DEC91    2.63312E9
                      23    SSE         DEC91    394258270
                      24    MSE         DEC91    3032755.9
                      25    RMSE        DEC91     1741.481
                      26    MAPE        DEC91    9.4800217
                      27    MPE         DEC91    -1.049956
                      28    MAE         DEC91    1306.8534
                      29    ME          DEC91    -42.95376
                      30    RSQUARE     DEC91    0.8502696
```

The following statements plot the residuals. The plot is shown in Output 17.1.4.

```
title2 'Plot of Residuals';
symbol1 i=needle;
axis2 label=( a=-90 r=90 "Vehicles and Parts" );
proc gplot data=out;
   plot vehicles * date = 1 / vref=0
        haxis= '1jan80'd to '1jan92'd by year
        vaxis=axis2;
   where _type_ = 'RESIDUAL';
   format date year4.;
run;
```

Output 17.1.4. Residuals from Winters Method

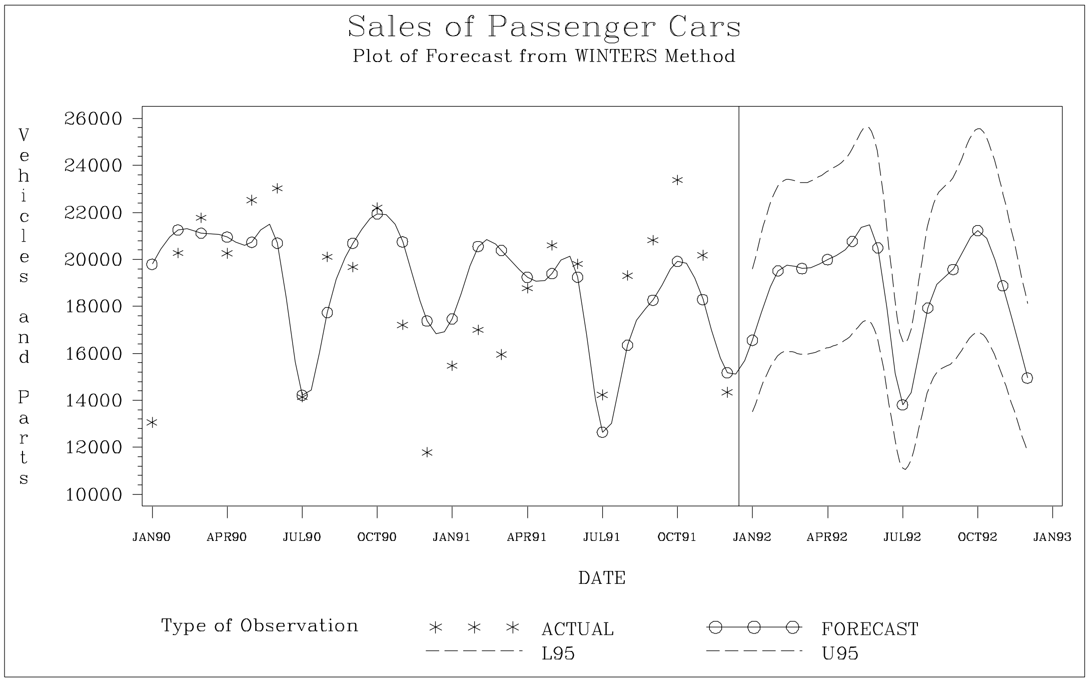


The following statements plot the forecast and confidence limits. The last two years of historical data are included in the plot to provide context for the forecast plot. A reference line is drawn at the start of the forecast period.

```
title2 ’Plot of Forecast from WINTERS Method’;
symbol1 i=none   v=star; /* for _type_=ACTUAL */
symbol2 i=spline v=circle; /* for _type_=FORECAST */
symbol3 i=spline l=3;        /* for _type_=L95 */
symbol4 i=spline l=3;        /* for _type_=U95 */
axis2 label=( a=-90 r=90 "Vehicles and Parts" )
      order=(6000 to 24000 by 3000) ;

proc gplot data=out;
   plot vehicles * date = _type_ /
        href= ’15dec91’d
        haxis= ’1jan90’d to ’1jan93’d by qtr
        vaxis=axis2;
   where _type_ ^= ’RESIDUAL’ & date >= ’1jan90’d;
run;
```

The plot is shown in Output 17.1.5.

Output 17.1.5. Forecast of Passenger Car Sales



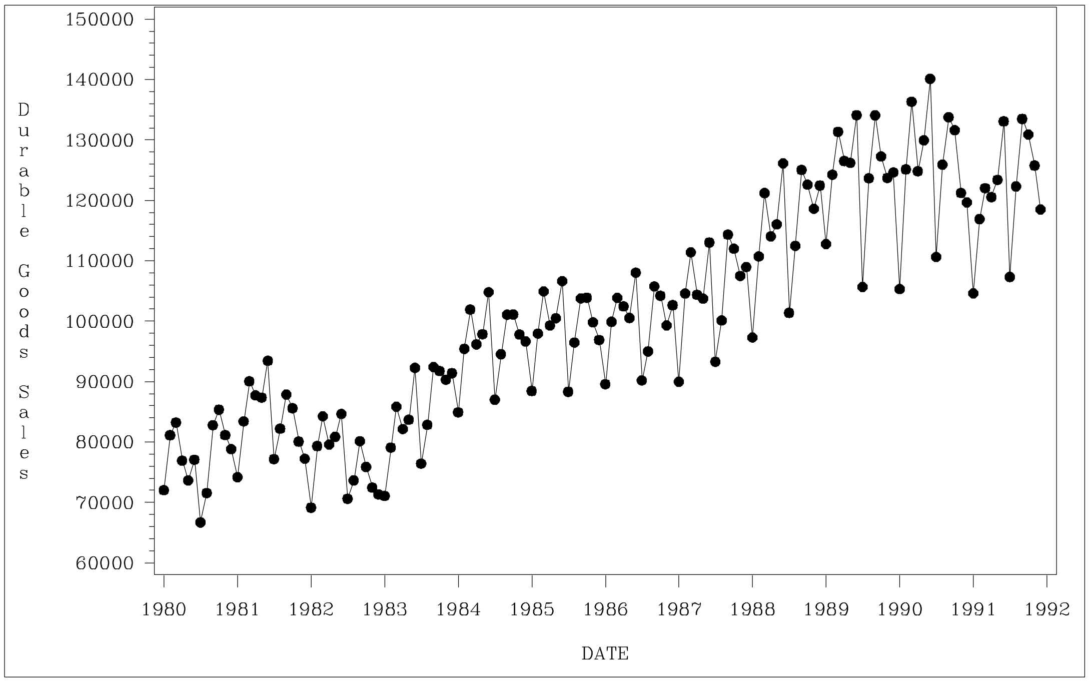
Example 17.2. Forecasting Retail Sales

This example uses the stepwise autoregressive method to forecast the monthly U.S. sales of durable goods (DURABLES) and nondurable goods (NONDUR) from the SASHELP.USECON data set. The data are from *Business Statistics* published by the U.S. Bureau of Economic Analysis. The following statements plot the series:

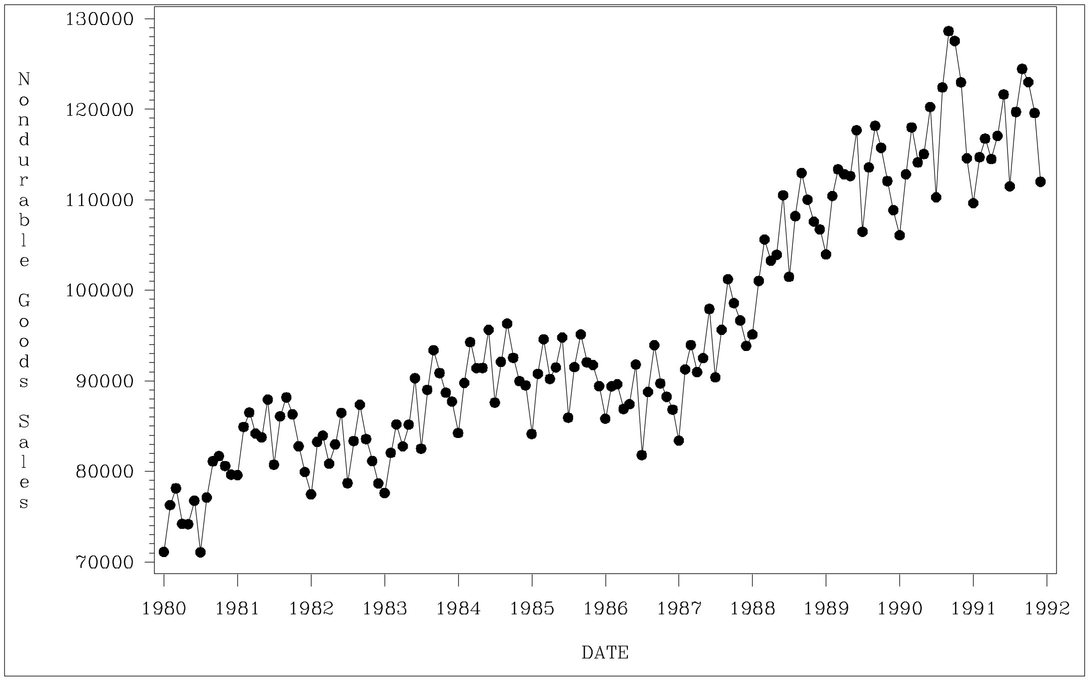
```
symbol1 i=spline v=dot;
proc gplot data=sashelp.usecon;
   plot ( durables nondur ) * date = 1 /
        haxis= '1jan80'd to '1jan92'd by year;
   where date >= '1jan80'd;
   format date year4.;
run;
```

The plots are shown in Output 17.2.1 and Output 17.2.2.

Output 17.2.1. Durable Goods Sales



Output 17.2.2. Nondurable Goods Sales



The following statements produce the forecast:

```
title1 "Forecasting Sales of Durable and Nondurable Goods";

proc forecast data=sashelp.usecon interval=month
              method=stepar trend=2 lead=12
              out=out outfull outest=est;
   id date;
   var durables nondur;
   where date >= '1jan80'd;
run;
```

The following statements print the OUTEST= data set.

```
title2 'OUTEST= Data Set: STEPAR Method';
proc print data=est;
run;
```

The PROC PRINT listing of the OUTEST= data set is shown in Output 17.2.3.

Output 17.2.3. The OUTEST= Data Set Produced by PROC FORECAST

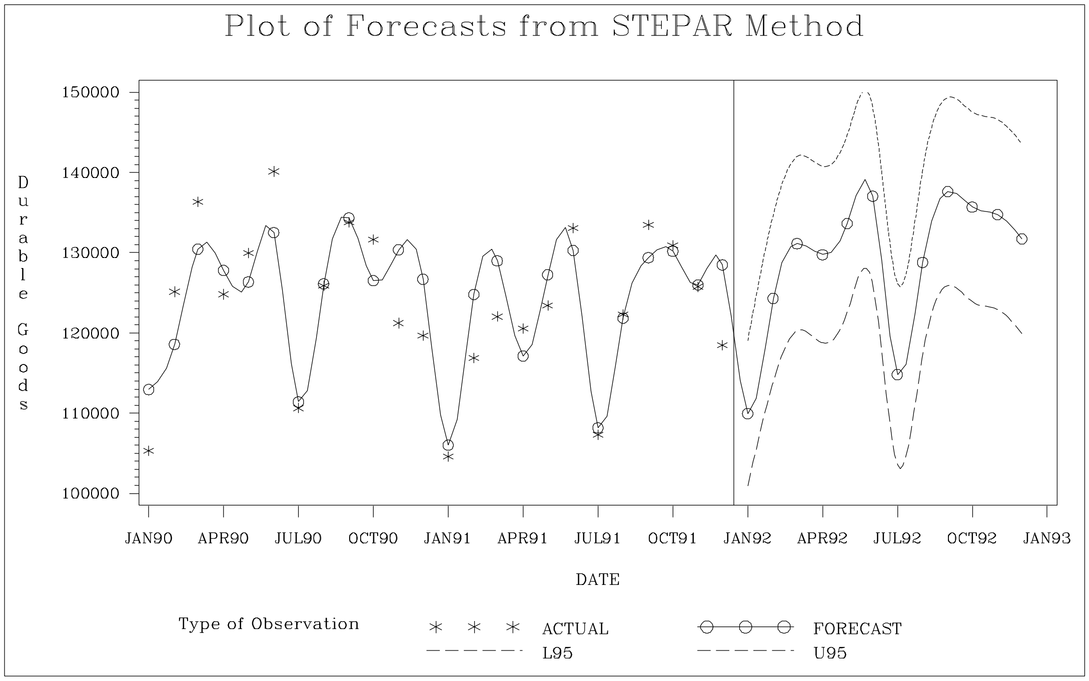
```
                Forecasting Sales of Durable and Nondurable Goods
                         OUTEST= Data Set: STEPAR Method

                Obs    _TYPE_      DATE       DURABLES       NONDUR

                  1    N           DEC91           144          144
                  2    NRESID      DEC91           144          144
                  3    DF          DEC91           137          139
                  4    SIGMA       DEC91      4519.451    2452.2642
                  5    CONSTANT    DEC91     71884.597    73190.812
                  6    LINEAR      DEC91     400.90106     308.5115
                  7    AR01        DEC91     0.5844515    0.8243265
                  8    AR02        DEC91             .            .
                  9    AR03        DEC91             .            .
                 10    AR04        DEC91             .            .
                 11    AR05        DEC91             .            .
                 12    AR06        DEC91     0.2097977            .
                 13    AR07        DEC91             .            .
                 14    AR08        DEC91             .            .
                 15    AR09        DEC91             .            .
                 16    AR10        DEC91     -0.119425            .
                 17    AR11        DEC91             .            .
                 18    AR12        DEC91     0.6138699    0.8050854
                 19    AR13        DEC91     -0.556707    -0.741854
                 20    SST         DEC91      4.923E10    2.8331E10
                 21    SSE         DEC91     1.88157E9    544657337
                 22    MSE         DEC91      13734093    3918398.1
                 23    RMSE        DEC91     3705.9538    1979.4944
                 24    MAPE        DEC91     2.9252601    1.6555935
                 25    MPE         DEC91     -0.253607    -0.085357
                 26    MAE         DEC91      2866.675    1532.8453
                 27    ME          DEC91     -67.87407    -29.63026
                 28    RSQUARE     DEC91     0.9617803    0.9807752
```

The following statements plot the forecasts and confidence limits. The last two years of historical data are included in the plots to provide context for the forecast. A reference line is drawn at the start of the forecast period.

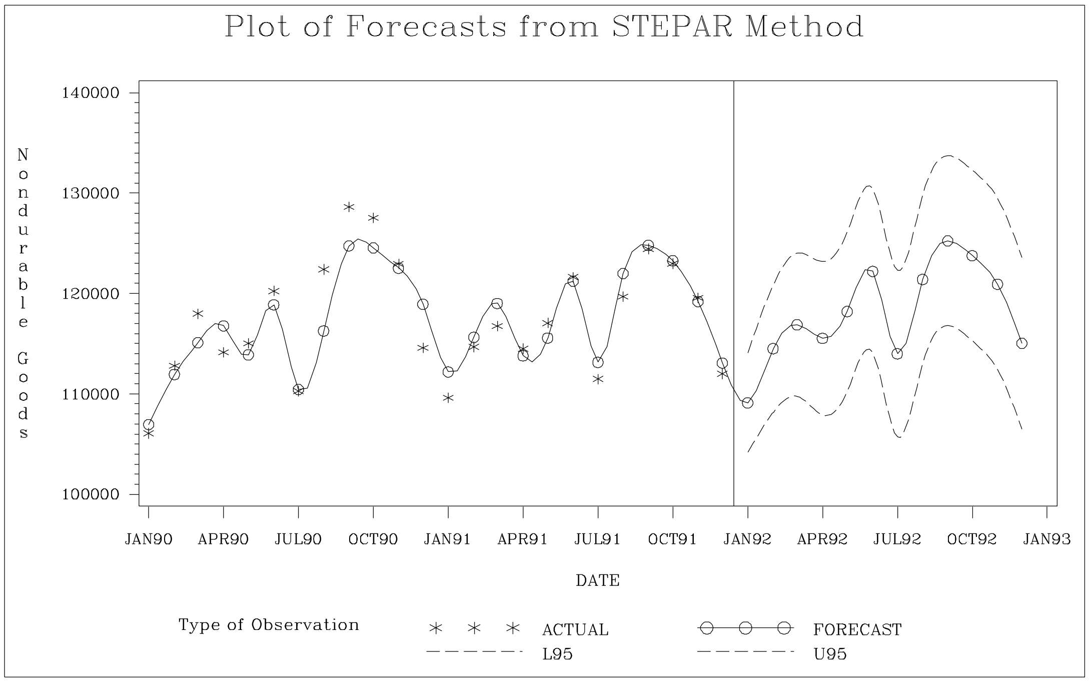
```
title1 'Plot of Forecasts from STEPAR Method';

symbol1 i=none   v=star h=2; /* for _type_=ACTUAL */
symbol2 i=spline v=circle h=2; /* for _type_=FORECAST */
symbol3 i=spline l=3;        /* for _type_=L95 */
symbol4 i=spline l=3;        /* for _type_=U95 */

proc gplot data=out;
   plot ( durables nondur ) * date = _type_ /
        href= '15dec91'd
        haxis= '1jan90'd to '1jan93'd by qtr;
   where date >= '1jan90'd;
run;
```

The plots are shown in Output 17.2.4 and Output 17.2.5.

Output 17.2.4. Forecast of Durable Goods Sales



Output 17.2.5. Forecast of Nondurable Goods Sales



Example 17.3. Forecasting Petroleum Sales

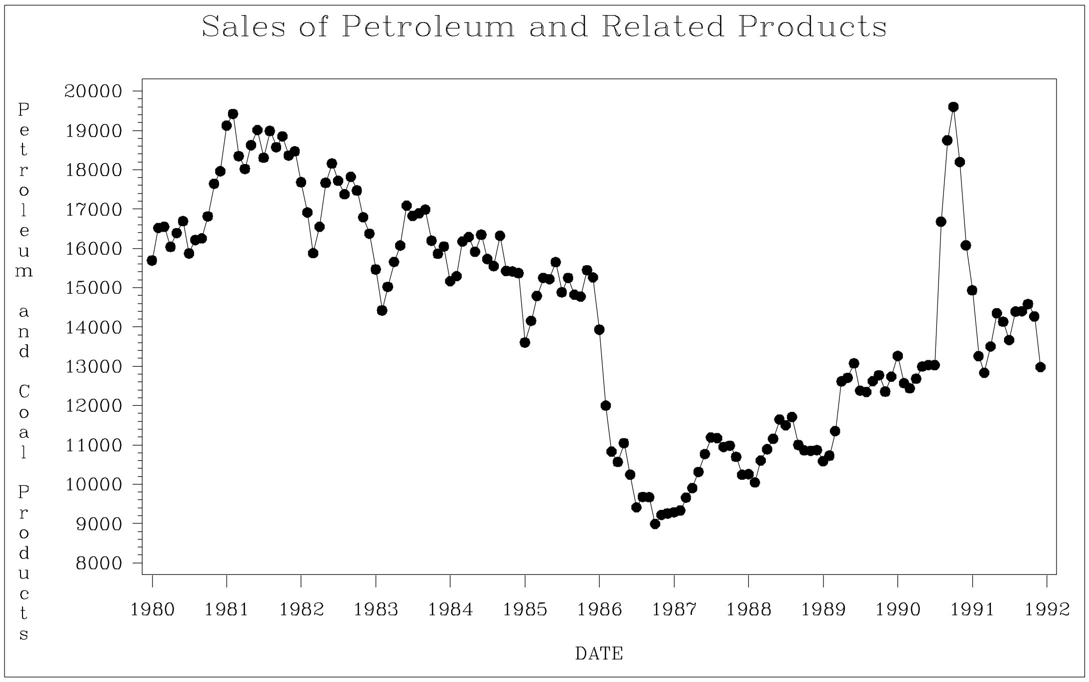
This example uses the double exponential smoothing method to forecast the monthly U.S. sales of petroleum and related products series (PETROL) from the data set SASHELP.USECON. These data are taken from *Business Statistics*, published by the U.S. Bureau of Economic Analysis.

The following statements plot the PETROL series:

```
title1 "Sales of Petroleum and Related Products";

symbol1 i=spline v=circle;
axis2 label=( a=-90 r=90 "Petroleum and Coal Products");
proc gplot data=sashelp.usecon;
   plot petrol * date = 1 /
        haxis= '1jan80'd to '1jan92'd by year
        vaxis=axis2;
   where date >= '1jan80'd;
   format date year4.;
run;
```

The plot is shown in Output 17.3.1.

Output 17.3.1. Sales of Petroleum and Related Products



The following statements produce the forecast:

```
proc forecast data=sashelp.usecon interval=month
              method=expo trend=2 lead=12
              out=out outfull outest=est;
   id date;
   var petrol;
   where date >= ’1jan80’d;
run;
```

The following statements print the OUTEST= data set:

```
title2 ’OUTEST= Data Set: EXPO Method’;
proc print data=est;
run;
```

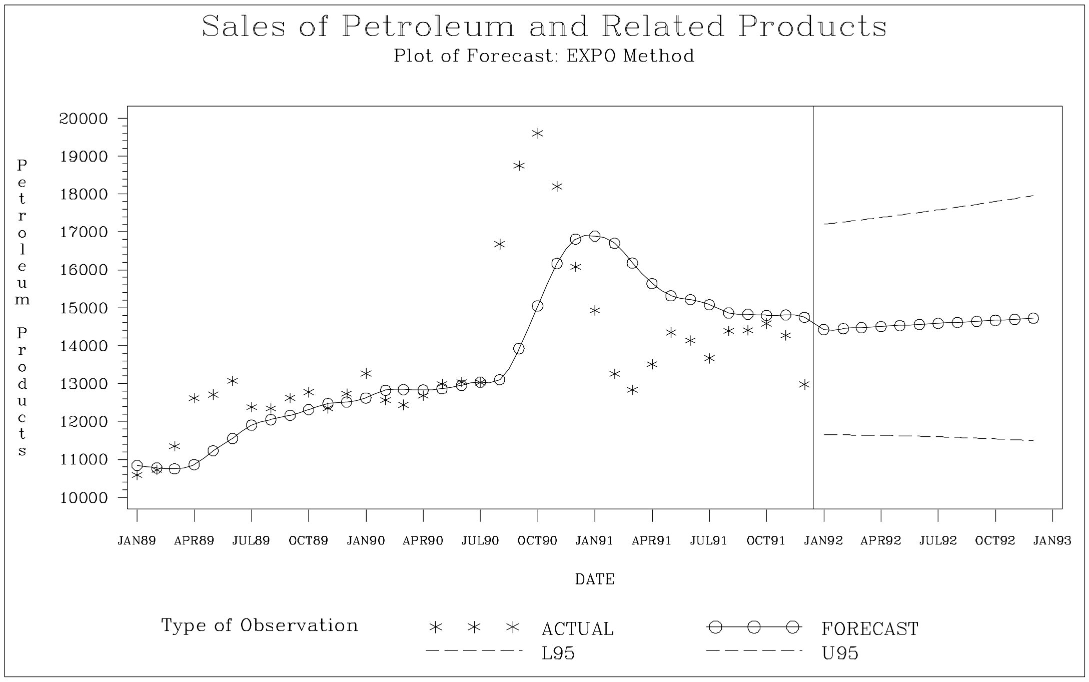
The PROC PRINT listing of the output data set is shown in Output 17.3.2.

Output 17.3.2. The OUTEST= Data Set Produced by PROC FORECAST

```
                    OUTEST= Data Set: EXPO Method

                Obs    _TYPE_       DATE         PETROL

                  1    N            DEC91           144
                  2    NRESID       DEC91           144
                  3    DF           DEC91           142
                  4    WEIGHT       DEC91     0.1055728
                  5    S1           DEC91     14165.259
                  6    S2           DEC91     13933.435
                  7    SIGMA        DEC91     1281.0945
                  8    CONSTANT     DEC91     14397.084
                  9    LINEAR       DEC91     27.363164
                 10    SST          DEC91     1.17001E9
                 11    SSE          DEC91     233050838
                 12    MSE          DEC91     1641203.1
                 13    RMSE         DEC91     1281.0945
                 14    MAPE         DEC91     6.5514467
                 15    MPE          DEC91     -0.147168
                 16    MAE          DEC91     891.04243
                 17    ME           DEC91     8.2148584
                 18    RSQUARE      DEC91     0.8008122
```

The plot of the forecast is shown in Output 17.3.3.

Output 17.3.3. Forecast of Petroleum and Related Products



References

Ahlburg, D. A. (1984). "Forecast evaluation and improvement using Theil's decomposition." *Journal of Forecasting*, 3, 345-351.

Aldrin, M. and Damsleth, E. (1989). "Forecasting non-seasonal time series with missing observations." *Journal of Forecasting*, 8, 97-116.

Archibald, B.C. (1990), "Parameter space of the Holt-Winters' model." *International Journal of Forecasting*, 6, 199-209.

Bails, D.G. and Peppers, L.C. (1982), *Business Fluctuations: Forecasting Techniques and Applications,* New Jersey: Prentice-Hall.

Bartolomei, S.M. and Sweet, A.L. (1989). "A note on the comparison of exponential smoothing methods for forecasting seasonal series." *International Journal of Forecasting*, 5, 111-116.

Bureau of Economic Analysis, U.S. Department of Commerce (1992 and earlier editions), *Business Statistics*, 27th and earlier editions, Washington: U.S. Government Printing Office.

Bliemel, F. (1973). "Theil's forecast accuracy coefficient: a clarification." *Journal of Marketing Research*, 10, 444-446.

Bowerman, B.L. and O'Connell, R.T. (1979), *Time Series and Forecasting: An Applied Approach,* North Scituate, Massachusetts: Duxbury Press.

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control,* Revised Edition, San Francisco: Holden-Day.

Bretschneider, S.I., Carbone, R., and Longini, R.L. (1979). "An adaptive approach to time series forecasting." *Decision Sciences*, 10, 232-244.

Brown, R.G. (1962), *Smoothing, Forecasting and Prediction of Discrete Time Series*, New York: Prentice-Hall.

Brown, R.G. and Meyer, R.F. (1961). "The fundamental theorem of exponential smoothing." *Operations Research*, 9, 673-685.

Chatfield, C. (1978). "The Holt-Winters forecasting procedure." *Applied Statistics*, **27**, 264-279.

Chatfield, C., and Prothero, D.L. (1973). "Box-Jenkins seasonal forecasting: problems in a case study." *Journal of the Royal Statistical Society, Series A*, 136, 295-315.

Chow, W.M. (1965). "Adaptive control of the exponential smoothing constant." *Journal of Industrial Engineering*, September-October 1965.

Cogger, K.O. (1974). "The optimality of general-order exponential smoothing." *Operations Research*, 22, 858-.

Cox, D. R. (1961). "Prediction by exponentially weighted moving averages and related methods." *Journal of the Royal Statistical Society, Series B*, 23, 414-422.

Fair, R.C. (1986). "Evaluating the predictive accuracy of models." In *Handbook of Econometrics*, Vol. 3., Griliches, Z. and Intriligator, M.D., eds. New York: North Holland.

Fildes, R. (1979). "Quantitative forecasting — the state of the art: extrapolative models." *Journal of Operational Research Society*, 30, 691-710.

Gardner, E.S. (1984). "The strange case of the lagging forecasts." *Interfaces*, 14, 47-50.

Gardner, E.S., Jr. (1985). "Exponential smoothing: the state of the art." *Journal of Forecasting*, 4, 1-38.

Granger, C.W.J. and Newbold, P. (1977), *Forecasting Economic Time Series*, New York: Academic Press, Inc.

Harvey, A.C. (1984). "A unified view of statistical forecasting procedures." *Journal of Forecasting*, 3, 245-275.

Ledolter, J. and Abraham, B. (1984). "Some comments on the initialization of exponential smoothing." *Journal of Forecasting*, 3, 79-84.

Maddala, G.S. (1977), *Econometrics*, New York: McGraw-Hill Book Co.

Makridakis, S., Wheelwright, S.C., and McGee, V.E. (1983). *Forecasting: Methods and Applications, 2nd Ed.* New York: John Wiley and Sons.

McKenzie, Ed (1984). "General exponential smoothing and the equivalent ARMA process." *Journal of Forecasting*, 3, 333-344.

Montgomery, D.C. and Johnson, L.A. (1976), *Forecasting and Time Series Analysis*, New York: McGraw-Hill Book Co.

Muth, J.F. (1960). "Optimal properties of exponentially weighted forecasts." *Journal of the American Statistical Association*, 55, 299-306.

Pierce, D.A. (1979). "R^2 Measures for Time Series." *Journal of the American Statistical Association*, 74, 901-910.

Pindyck, R.S. and Rubinfeld, D.L. (1981), *Econometric Models and Economic Forecasts*, Second Edition, New York: McGraw-Hill Book Co.

Raine, J.E. (1971). "Self-adaptive forecasting reconsidered." *Decision Sciences*, 2, 181-191.

Roberts, S.A. (1982). "A general class of Holt-Winters type forecasting models." *Management Science*, 28, 808-820.

Theil, H. (1966). *Applied Economic Forecasting*. Amsterdam: North Holland.

Trigg, D.W., and Leach, A.G. (1967). "Exponential smoothing with an adaptive response rate." *Operational Research Quarterly*, 18, 53-59.

Winters, P.R. (1960). "Forecasting sales by exponentially weighted moving averages." *Management Science*, 6, 324-342.

Chapter 18

The LOAN Procedure

Chapter Contents

Chapter 18
The LOAN Procedure

Overview

The LOAN procedure analyzes and compares fixed rate, adjustable rate, buydown, and balloon payment loans. The LOAN procedure computes the loan parameters and outputs the loan summary information for each loan.

Multiple loans can be processed and compared in terms of economic criteria such as after-tax or before-tax present worth of cost and true interest rate, breakeven of periodic payment and of interest paid, and outstanding balance at different periods in time. PROC LOAN selects the best alternative in terms of the specified economic criterion for each loan comparison period.

The LOAN procedure allows various payment and compounding intervals (including continuous compounding) and uniform or lump sum prepayments for each loan. Down payments, discount points, and other initialization costs can be included in the loan analysis and comparison.

Getting Started

PROC LOAN supports four types of loans. You specify each type of loan using the corresponding statement: FIXED, BALLOON, ARM, and BUYDOWN.

- FIXED - Fixed rate loans have a constant interest rate and periodic payment throughout the life of the loan.
- BALLOON - Balloon payment loans are fixed rate loans with lump sum payments in certain payment periods in addition to the constant periodic payment.
- ARM - Adjustable rate loans are those in which the interest rate and periodic payment vary over the life of the loan. The future interest rates of an adjustable rate loan are not known with certainty, but they will vary within specified limits according to terms stated in the loan agreement. In practice, the rate adjustment terms vary. PROC LOAN offers a flexible set of options to capture a wide variety of rate adjustment terms.
- BUYDOWN - Buydown rate loans are similar to adjustable rate loans, but the interest rate adjustments are predetermined at the initialization of the loan, usually by paying interest points at the time of loan initialization.

Analyzing Fixed Rate Loans

The most common loan analysis is the calculation of the periodic payment when the loan amount, life, and interest rate are known. The following PROC LOAN statements analyze a 15-year (180 monthly payments) fixed rate loan for $100,000 with an annual nominal interest rate of 7.5%:

```
proc loan;
   fixed amount=100000 rate=7.5 life=180;
run;
```

Another parameter PROC LOAN can compute is the maximum amount you can borrow given the periodic payment you can afford and the rates available in the market. The following SAS statements analyze a loan for 180 monthly payments of $900, with a nominal annual rate of 7.5%:

```
proc loan;
   fixed payment=900 rate=7.5 life=180;
run;
```

Assume that you want to borrow $100,000 and can pay $900 a month. You know that the lender charges a 7.5% nominal interest rate compounded monthly. To determine how long it will take you to pay off your debt, use the following statements:

```
proc loan;
   fixed amount=100000 payment=900 rate=7.5;
run;
```

Sometimes, a loan is expressed in terms of the amount borrowed and the amount and number of periodic payments. In this case, you want to calculate the annual nominal rate charged on the loan to compare it to other alternatives. The following statements analyze a loan of $100,000 paid in 180 monthly payments of $800:

```
proc loan;
   fixed amount=100000 payment=800 life=180;
run;
```

There are four basic parameters that define a loan: life (number of periodic payments), principal amount, interest rate, and the periodic payment amount. PROC LOAN calculates the missing parameter among these four. Loan analysis output includes a loan summary table and an amortization schedule.

You can use the START= and LABEL= options to enhance your output. The START= option specifies the date of loan initialization and dates all the output accordingly. The LABEL= specification is used to label all output corresponding to a particular loan and is especially useful when multiple loans are analyzed. For example, the preceding statements for the first fixed rate loan are revised to include the START= and LABEL= options as follows:

```
proc loan start=1998:12;
   fixed amount=100000 rate=7.5 life=180
         label='BANK1, Fixed Rate';
run;
```

Loan Summary Table

The loan summary table is produced by default and contains loan analysis information. It shows the principal amount, the costs at the time of loan initialization (downpayment, discount points, and other loan initialization costs), the total payment and interest, the initial nominal and effective interest rates, payment and compounding intervals, the length of the loan in the time units specified, the start and end dates (if specified), a list of nominal and effective interest rates, and periodic payments throughout the life of the loan.

Figure 18.1 shows the loan summary table for the fixed rate loan labeled "BANK1, Fixed Rate."

```
                                 The LOAN Procedure

                             Fixed Rate Loan Summary

                                  BANK1, Fixed Rate

   Downpayment                  0.00    Principal Amount           100000.00
   Initialization               0.00    Points                          0.00
   Total Interest           66862.61    Nominal Rate                 7.5000%
   Total Payment           166862.61    Effective Rate               7.7633%
   Pay Interval              MONTHLY    Compounding                  MONTHLY
   No. of Payments               180    No. of Compoundings              180
   Start Date                DEC1998    End Date                     DEC2013


                    Rates and Payments for BANK1, Fixed Rate
        Date            Nominal Rate     Effective Rate         Payment

        DEC1998            7.5000%            7.7633%            927.01
```

Figure 18.1. Fixed Rate Loan Summary

The loan is initialized in December 1998 and paid off in December 2013. The monthly payment is calculated to be $927.01, and the effective interest rate is 7.7633%. Over the 15 years, $66,862.61 is paid for interest charges on the loan.

Analyzing Balloon Payment Loans

You specify balloon payment loans like fixed rate loans, with the additional specification of the balloon payments. Assume you have an alternative to finance the $100,000 investment with a 15-year balloon payment loan. The annual nominal rate is 7.5%, as in the fixed rate loan. The terms of the loan require two balloon payments of $2000 and $1000 at the 15th and 48th payment periods, respectively. These balloon payments keep the periodic payment lower than that of the fixed rate loan. The balloon payment loan is defined by the following BALLOON statement:

```
proc loan start=1998:12;
   balloon amount=100000 rate=7.5 life=180
           balloonpayment=(15=2000 48=1000)
           label = 'BANK2, with Balloon Payment';
run;
```

List of Balloon Payments

In addition to the information for the fixed rate loan, the “Loan Summary Table” for the balloon payment loan includes a list of balloon payments in the “List of Rates and Payments.” For example, the balloon payment loan described previously includes two balloon payments, as shown in Figure 18.2.

```
                                The LOAN Procedure

                  Rates and Payments for BANK2, with Balloon Payment
             Date           Nominal Rate      Effective Rate          Payment

             DEC1998            7.5000%            7.7633%            903.25



                            Balloon Period          Payment

                            MAR2000                 2000.00
                            DEC2002                 1000.00
```

Figure 18.2. List of Rates and Payments for a Balloon Payment Loan

The periodic payment for the balloon payment loan is $23.76 less than that of the fixed rate loan.

Analyzing Adjustable Rate Loans

In addition to specifying the basic loan parameters, you need to specify the terms of the rate adjustments for an adjustable rate loan. There are many ways of stating the rate adjustment terms, and PROC LOAN facilitates all of them. For details, see the “Rate Adjustment Terms Options” in the “ARM Statement” section later in this chapter.

Assume that you have an alternative to finance the $100,000 investment with a 15-year adjustable rate loan with an initial annual nominal interest rate of 5.5%. The rate adjustment terms specify a 0.5% annual cap, a 2.5% life cap, and a rate adjustment every 12 months. *Annual cap* refers to the maximum increase in interest rate per adjustment period, and *life cap* refers to the maximum increase over the life of the loan. The following ARM statement specifies this adjustable rate loan assuming the interest rate adjustments will always increase by the maximum allowed by the terms of the loan. These assumptions are specified by the WORSTCASE and CAPS= options, as shown in the following statements:

```
proc loan start=1998:12;
   arm amount=100000 rate=5.5 life=180 worstcase
       caps=(0.5, 2.5)
       label='BANK3, Adjustable Rate';
run;
```

List of Rates and Payments for Adjustable Rate Loans

The “List of Rates and Payments” in the loan summary table for the adjustable rate loans reflects the changes in the interest rates and payments, as well as the dates these changes become effective. For the adjustable rate loan described previously, Figure 18.3 shows the “List of Rates and Payments” indicating five annual rate adjustments in addition to the initial rate and payment.

```
                              The LOAN Procedure

                  Rates and Payments for BANK3, Adjustable Rate
      Date            Nominal Rate      Effective Rate          Payment

      DEC1998             5.5000%             5.6408%            817.08
      JAN2000             6.0000%             6.1678%            842.33
      JAN2001             6.5000%             6.6972%            866.44
      JAN2002             7.0000%             7.2290%            889.32
      JAN2003             7.5000%             7.7633%            910.88
      JAN2004             8.0000%             8.3000%            931.03
```

Figure 18.3. List of Rates and Payments for an Adjustable Rate Loan

Notice that the periodic payment of the adjustable rate loan as of January 2004 ($931.03) exceeds that of the fixed rate loan ($927.01).

Analyzing Buydown Rate Loans

A 15-year buydown rate loan is another alternative to finance the $100,000 investment. The nominal annual interest rate is 6.5% initially and will increase to 8% and 9% as of the 24th and 48th payment periods, respectively. The nominal annual interest rate is lower than that of the fixed rate alternative, at the cost of a 1% discount point ($1000) paid at the initialization of the loan. The following BUYDOWN statement represents this loan alternative:

```
proc loan start=1998:12;
   buydown amount=100000 rate=6.5 life=180
           buydownrates=(24=8 48=9) pointpct=1
           label='BANK4, Buydown';
run;
```

List of Rates and Payments for Buydown Rate Loans

Figure 18.4 shows the “List of Rates and Payments” in the loan summary table. It reflects the two rate adjustments and the corresponding monthly payments as well as the initial values for these parameters. As of December 2000, the periodic payment of the buydown loan exceeds the periodic payment for any of the other alternatives.

```
                              The LOAN Procedure

                      Rates and Payments for BANK4, Buydown
      Date            Nominal Rate      Effective Rate          Payment

      DEC1998             6.5000%             6.6972%            871.11
      DEC2000             8.0000%             8.3000%            946.50
      DEC2002             9.0000%             9.3807%            992.01
```

Figure 18.4. List of Rates and Payments for a Buydown Rate Loan

Loan Repayment Schedule

In addition to the loan summary, you can print a loan repayment (amortization) schedule for each loan. For each payment period, this schedule contains the year and period within the year (or date, if the START= option is specified), the principal balance at the beginning of the period, the total payment, interest payment, principal repayment for the period, and the principal balance at the end of the period.

To print the first year of the amortization schedule for the fixed rate loan shown in Figure 18.5, use the following statements:

```
proc loan start=1998:12;
   fixed amount=100000 rate=7.5 life=180
         schedule=1
         label='BANK1, Fixed Rate';
run;
```

```
                                The LOAN Procedure

                             Loan Repayment Schedule

                                BANK1, Fixed Rate

               Beginning                       Interest      Principal         Ending
Date         Outstanding         Payment        Payment      Repayment    Outstanding

DEC1998        100000.00            0.00           0.00           0.00      100000.00

DEC1998        100000.00            0.00           0.00           0.00      100000.00

JAN1999        100000.00          927.01         625.00         302.01       99697.99
FEB1999         99697.99          927.01         623.11         303.90       99394.09
MAR1999         99394.09          927.01         621.21         305.80       99088.29
APR1999         99088.29          927.01         619.30         307.71       98780.58
MAY1999         98780.58          927.01         617.38         309.63       98470.95
JUN1999         98470.95          927.01         615.44         311.57       98159.38
JUL1999         98159.38          927.01         613.50         313.51       97845.87
AUG1999         97845.87          927.01         611.54         315.47       97530.40
SEP1999         97530.40          927.01         609.57         317.44       97212.96
OCT1999         97212.96          927.01         607.58         319.43       96893.53
NOV1999         96893.53          927.01         605.58         321.43       96572.10
DEC1999         96572.10          927.01         603.58         323.43       96248.67

DEC1999        100000.00        11124.12        7372.79        3751.33       96248.67
```

Figure 18.5. Loan Repayment Schedule for the First Year

The principal balance at the end of one year is $96,248.67. The total payment for the year is $11,124.12 of which $3,751.33 went toward principal repayment.

You can also print the amortization schedule with annual summary information or for a specified number of years. The SCHEDULE=YEARLY option produces an annual summary loan amortization schedule, which is useful for loans with long life. For example, to print the annual summary loan repayment schedule for the buydown loan shown in Figure 18.6, use the following statements.

```
proc loan start=1998:12;
   buydown amount=100000 rate=6.5 life=180
           buydownrates=(24=8 48=9) pointpct=1
           schedule=yearly
           label='BANK4, Buydown';
run;
```

```
                                  The LOAN Procedure

                                Loan Repayment Schedule

                                    BANK4, Buydown

            Beginning                        Interest       Principal          Ending
   Year    Outstanding        Payment         Payment       Repayment     Outstanding

   1998      100000.00        1000.00            0.00            0.00       100000.00
   1999      100000.00       10453.32         6380.07         4073.25        95926.75
   2000       95926.75       10528.71         6222.21         4306.50        91620.25
   2001       91620.25       11358.00         7178.57         4179.43        87440.82
   2002       87440.82       11403.51         6901.12         4502.39        82938.43
   2003       82938.43       11904.12         7276.64         4627.48        78310.95
   2004       78310.95       11904.12         6842.58         5061.54        73249.41
   2005       73249.41       11904.12         6367.76         5536.36        67713.05
   2006       67713.05       11904.12         5848.43         6055.69        61657.36
   2007       61657.36       11904.12         5280.35         6623.77        55033.59
   2008       55033.59       11904.12         4659.00         7245.12        47788.47
   2009       47788.47       11904.12         3979.34         7924.78        39863.69
   2010       39863.69       11904.12         3235.96         8668.16        31195.53
   2011       31195.53       11904.12         2422.83         9481.29        21714.24
   2012       21714.24       11904.12         1533.41        10370.71        11343.53
   2013       11343.53       11904.09          560.56        11343.53            0.00
```

Figure 18.6. Annual Summary Loan Repayment Schedule

Loan Comparison

The LOAN procedure can compare alternative loans on the basis of different economic criteria and help select the most desirable loan. You can compare alternative loans through different points in time. The economic criteria offered by PROC LOAN are

- outstanding principal balance, that is, the unpaid balance of the loan
- present worth of cost, that is, before-tax or after-tax net value of the loan cash flow through the comparison period. The cash flow includes all payments, discount points, initialization costs, down payment, and the outstanding principal balance at the comparison period.
- true interest rate, that is, before-tax or after-tax effective annual interest rate charged on the loan. The cash flow includes all payments, discount points, initialization costs, and the outstanding principal balance at the specified comparison period.
- periodic payment
- the total interest paid on the loan

The figures for present worth of cost, true interest rate, and interest paid are reported on the cash flow through the comparison period. The reported outstanding principal balance and the periodic payment are the values as of the comparison period.

The COMPARE statement specifies the type of comparison and the periods of comparison. For each period specified in the COMPARE statement, a loan comparison report is printed that also indicates the best alternative. Different criteria may lead to selection of different alternatives. Also, the period of comparison may change the desirable alternative. See the section "Loan Comparison Details" later in this chapter for further information.

Comparison of 15-Year versus 30-Year Loan Alternatives

An issue that arises in the purchase of a house is the length of the loan life. In the U.S., residential home loans are usually for 15 or 30 years. Ordinarily, 15-year loans have a lower interest rate but higher periodic payments than 30-year loans. A comparison of both loans might identify the better loan for your means and needs. The following SAS statements compare two such loans:

```
proc loan start=1998:12 amount=120000;
   fixed rate=7.5 life=360 label='30 year loan';
   fixed rate=6.5 life=180 label='15 year loan';
   compare;
run;
```

Default Loan Comparison Report

The default loan comparison report in Figure 18.7 shows the ending outstanding balance, periodic payment, interest paid, and before-tax true rate at the end of 30 years. In the case of the default loan comparison, the selection of the best alternative is based on minimization of the true rate.

```
                              The LOAN Procedure

                            Loan Comparison Report

                          Analysis through DEC2028

                               Ending                      Interest       True
  Loan Label              Outstanding        Payment          Paid        Rate

  30 year loan                   0.00         835.48     182058.02        7.76
  15 year loan                   0.00        1044.95      68159.02        6.70

NOTE: "15 year loan" is the best alternative based on true rate analysis through DEC2028.
```

Figure 18.7. Default Loan Comparison Report

Based on true rate, the best alternative is the 15-year loan. However, if the objective were to minimize the periodic payment, the 30-year loan would be the more desirable.

Comparison of Fixed Rate and Adjustable Rate Loans

Suppose you want to compare a fixed rate loan to an adjustable rate alternative. The nominal interest rate on the adjustable rate loan is initially 1.5% lower than the fixed rate loan. The future rates of the adjustable rate loan are calculated using the worst case scenario.

According to current U.S. tax laws, the loan for a family home qualifies the interest paid on the loan as a tax deduction. The TAXRATE=28 (income tax rate) option on the COMPARE statement bases the calculations of true interest rate on the after-tax cash flow. Assume, also, that you are uncertain as to how long you will keep this property. The AT=(60 120) option, as shown in the following example, produces two loan comparison reports through the end of the 5th and the 10th years, respectively:

```
proc loan start=1998:12 amount=120000 life=360;
   fixed rate=7.5 label='BANK1, Fixed Rate';
   arm   rate=6.0 worstcase caps=(0.5 2.5)
         label='BANK3, Adjustable Rate';
   compare taxrate=28 at=(60 120);
run;
```

After-Tax Loan Comparison Reports

The two loan comparison reports in Figure 18.8 and Figure 18.9 show the ending outstanding balance, periodic payment, interest paid, and after-tax true rate at the end of five years and ten years, respectively.

```
                                  The LOAN Procedure

                                Loan Comparison Report

                              Analysis through DEC2003

                                   Ending                         Interest       True
   Loan Label                  Outstanding          Payment           Paid       Rate

   BANK1, Fixed Rate             113540.74           839.06       43884.34       5.54
   BANK3, Adjustable Rate        112958.49           871.83       40701.93       5.11

NOTE: "BANK3, Adjustable Rate" is the best alternative based on true rate analysis through
                                       DEC2003.
```

Figure 18.8. Loan Comparison Report as of December 2003

```
                                  The LOAN Procedure

                                Loan Comparison Report

                              Analysis through DEC2008

                                   Ending                         Interest       True
   Loan Label                  Outstanding          Payment           Paid       Rate

   BANK1, Fixed Rate             104153.49           839.06       84840.69       5.54
   BANK3, Adjustable Rate        104810.98           909.57       87128.62       5.60

NOTE: "BANK1, Fixed Rate" is the best alternative based on true rate analysis through DEC2008.
```

Figure 18.9. Loan Comparison Report as of December 2008

The loan comparison report through December 2003 picks the adjustable rate loan as the best alternative, whereas the report through December 2008 shows the fixed rate loan as the better alternative. This implies that if you intend to keep the loan for 10 years or longer, the best alternative is the fixed rate alternative. Otherwise, the adjustable rate loan is the better alternative in spite of the worst-case scenario. Further analysis shows that the actual breakeven of true interest rate occurs at August 2008. That is, the desirable alternative switches from the adjustable rate loan to the fixed rate loan in August 2008.

Note that, under the assumption of worst-case scenario for the rate adjustments, the periodic payment for the adjustable rate loan already exceeds that of the fixed rate loan on December 2003 (as of the rate adjustment on January 2003 to be exact). If the objective were to minimize the periodic payment, the fixed rate loan would have been more desirable as of December 2003. However, all of the other criteria at that point still favor the adjustable rate loan.

Syntax

The following statements are used with PROC LOAN:

PROC LOAN *options* **;**
  FIXED *options* **;**
  BALLOON *options* **;**
  ARM *options* **;**
  BUYDOWN *options* **;**
  COMPARE *options* **;**

Functional Summary

The statements and options controlling the LOAN procedure are summarized in the following table. Many of the loan specification options can be used on all of the statements except the COMPARE statement. For these options, the statement column will be left blank. Options specific to a type of loan will indicate the statement name.

Description	Statement	Option
Statements		
specify an adjustable rate loan	ARM	
specify a balloon payment loan	BALLOON	
specify a buydown rate loan	BUYDOWN	
specify loan comparisons	COMPARE	
specify a fixed rate loan	FIXED	

Description	Statement	Option
Data Set Options		
specify output data set for loan summary	PROC LOAN	OUTSUM=
specify output data set for repayment schedule		OUT=
specify output data set for loan comparison	COMPARE	OUTCOMP=
Printing Control Options		
suppress printing of loan summary report		NOSUMMARYPRINT
suppress all printed output		NOPRINT
print amortization schedule		SCHEDULE=
suppress printing of loan comparison report	COMPARE	NOCOMPRINT
Required Specifications		
specify the loan amount		AMOUNT=
specify life of loan as number of payments		LIFE=
specify the periodic payment		PAYMENT=
specify the initial annual nominal interest rate		RATE=
Loan Specifications Options		
specify loan amount as percentage of price		AMOUNTPCT=
specify time interval between compoundings		COMPOUND=
specify down payment at loan initialization		DOWNPAYMENT=
specify down payment as percentage of price		DOWNPAYPCT=
specify amount paid for loan initialization		INITIAL=
specify initialization costs as a percent		INITIALPCT=
specify time interval between payments		INTERVAL=
specify label for the loan		LABEL=
specify amount paid for discount points		POINTS=
specify discount points as a percent		POINTPCT=
specify uniform or lump sum prepayments		PREPAYMENTS=
specify the purchase price		PRICE=
specify number of decimal places for rounding		ROUND=
specify the date of loan initialization		START=
Balloon Payment Loan Specification Option		
specify the list of balloon payments	BALLOON	BALLOONPAYMENT=
Rate Adjustment Terms Options		
specify frequency of rate adjustments	ARM	ADJUSTFREQ=
specify periodic and life cap on rate adjustment	ARM	CAPS=
specify maximum rate adjustment	ARM	MAXADJUST=
specify maximum annual nominal interest rate	ARM	MAXRATE=
specify minimum annual nominal interest rate	ARM	MINRATE=

Description	Statement	Option
Rate Adjustment Case Options		
specify best-case (optimistic) scenario	ARM	BESTCASE
specify predicted interest rates	ARM	ESTIMATEDCASE=
specify constant rate	ARM	FIXEDCASE
specify worst case (pessimistic) scenario	ARM	WORSTCASE
Buydown Rate Loan Specification Option		
specify list of nominal interest rates	BUYDOWN	BUYDOWNRATES=
Loan Comparison Options		
specify all comparison criteria	COMPARE	ALL
specify the loan comparison periods	COMPARE	AT=
specify breakeven analysis of the interest paid	COMPARE	BREAKINTEREST
specify breakeven analysis of periodic payment	COMPARE	BREAKPAYMENT
specify minimum attractive rate of return	COMPARE	MARR=
specify present worth of cost analysis	COMPARE	PWOFCOST
specify the income tax rate	COMPARE	TAXRATE=
specify true interest rate analysis	COMPARE	TRUEINTEREST

PROC LOAN Statement

PROC LOAN *options ;*

The following output opton can be used in the PROC LOAN statement. In addition, the following loan specification options can be specified in the PROC LOAN statement to be used as defaults for all loans unless otherwise specified for a given loan:

AMOUNT=	INTERVAL=	POINTPCT=
AMOUNTPCT=	LABEL=	PREPAYMENTS=
COMPOUND=	LIFE=	PRICE=
DOWNPAYMENT=	NOSUMMARYPRINT	RATE=
DOWNPAYPCT=	NOPRINT	ROUND=
INITIAL=	PAYMENT=	START=
INITIALPCT=	POINTS=	SCHEDULE=.

Output Option

OUTSUM= *SAS-data-set*
: creates an output data set containing loan summary information for all loans other than those for which a different OUTSUM= output data set is specified.

FIXED Statement

FIXED *options ;*

The FIXED statement specifies a fixed rate and periodic payment loan. It can be specified using the options that are common to all loan statements. The FIXED statement options are listed in this section.

You must specify three of the following options in each loan statement: AMOUNT=, LIFE=, RATE=, and PAYMENT=. The LOAN procedure calculates the fourth parameter based on the values you give the other three. If you specify all four of the options, the PAYMENT= specification is ignored, and the periodic payment is recalculated for consistency.

As an alternative to specifying the AMOUNT= option, you can specify the PRICE= option along with one of the following options to facilitate the calculation of the loan amount: AMOUNTPCT=, DOWNPAYMENT=, or DOWNPAYPCT=.

Required Specifications

AMOUNT= *amount*
A= *amount*
: specifies the loan amount (the outstanding principal balance at the initialization of the loan).

LIFE= *n*
L= *n*
: gives the life of the loan in number of payments. (The payment frequency is specified by the INTERVAL= option.) For example, if the life of the loan is 10 years with monthly payments, use LIFE=120 and INTERVAL=MONTH (default) to indicate a 10-year loan in which 120 monthly payments are made.

PAYMENT= *amount*
P= *amount*
: specifies the periodic payment. For ARM and BUYDOWN loans where the periodic payment might change, the PAYMENT= option specifies the initial amount of the periodic payment.

RATE= *rate*
R= *rate*
: specifies the initial annual (nominal) interest rate in percent notation. The rate specified must be in the range 0% to 120%. For example, use RATE=12.75 for a 12.75% loan. For ARM and BUYDOWN loans, where the rate might change over the life of the loan, the RATE= option specifies the initial annual interest rate.

Specification Options

AMOUNTPCT= *value*
APCT= *value*

specifies the loan amount as a percentage of the purchase price (PRICE= option). The AMOUNTPCT= specification is used to calculate the loan amount if the AMOUNT= option is not specified. The value specified must be in the range 1% to 100%.

If both the AMOUNTPCT= and DOWNPAYPCT= options are specified and the sum of their values is not equal to 100, the value of the downpayment percentage is set equal to 100 minus the value of the amount percentage.

COMPOUND= *time-unit*

specifies the time interval between compoundings. The default is the time unit given by the INTERVAL= option. If the INTERVAL= option is not used, then the default is COMPOUND=MONTH. The following time units are valid COMPOUND= values: CONTINUOUS, DAY, SEMIMONTH, MONTH, QUARTER, SEMIYEAR, and YEAR. The compounding interval is used to calculate the simple interest rate per payment period from the nominal annual interest rate or vice versa.

DOWNPAYMENT= *amount*
DP= *amount*

specifies the down payment at the initialization of the loan. The down payment is included in the calculation of the present worth of cost but not in the calculation of the true interest rate. The after-tax analysis assumes that the down payment is not tax-deductible. (Specify after-tax analysis with the TAXRATE= option in the COMPARE statement.)

DOWNPAYPCT= *value*
DPCT= *value*

specifies the down payment as a percentage of the purchase price (PRICE= option). The DOWNPAYPCT= specification is used to calculate the down payment amount if you do not specify the DOWNPAYMENT= option. The value you specify must be in the range 0% to 99%.

If you specified both the AMOUNTPCT= and DOWNPAYPCT= options, and the sum of their values is not equal to 100, the value of the downpayment percentage is set equal to 100 minus the value of the amount percentage.

INITIAL= *amount*
INIT= *amount*

specifies the amount paid for loan initialization other than the discount points and down payment. This amount is included in the calculation of the present worth of cost and the true interest rate. The after-tax analysis assumes that the initial amount is not tax-deductible. (After-tax analysis is specified by the TAXRATE= option in the COMPARE statement.)

INITIALPCT= *value*
INITPCT= *value*

specifies the initialization costs as a percentage of the loan amount (AMOUNT= option). The INITIALPCT= specification is used to calculate the amount paid for loan initialization if you do not specify the INITIAL= option. The value you specify must be in the range of 0% to 100%.

INTERVAL= *time-unit*
gives the time interval between periodic payments. The default is INTERVAL=MONTH. The following time units are valid INTERVAL values: SEMIMONTH, MONTH, QUARTER, SEMIYEAR, and YEAR.

LABEL= *'loan-label'*
specifies a label for the loan. If you specify the LABEL= option, all output related to the loan is labeled accordingly. If you do not specify the LABEL= option, the loan is labeled by sequence number.

POINTS= *amount*
PNT= *amount*
specifies the amount paid for discount points at the initialization of the loan. This amount is included in the calculation of the present worth of cost and true interest rate. The amount paid for discount points is assumed to be tax-deductible in after-tax analysis (that is, if the TAXRATE= option is specified in the COMPARE statement).

POINTPCT= *value*
PNTPCT= *value*
specifies the discount points as a percentage of the loan amount (AMOUNT= option). The POINTPCT= specification is used to calculate the amount paid for discount points if you do not specify the POINTS= option. The value you specify must be in the range of 0% to 100%.

PREPAYMENTS= *amount*
PREPAYMENTS= *(date1=prepayment1 date2=prepayment2 ...)*
PREPAYMENTS= *(period1=prepayment1 period2=prepayment2 ...)*
PREP=
specifies either a uniform prepayment p throughout the life of the loan or lump sum prepayments. A uniform prepayment, p, is assumed to be paid with each periodic payment. Specify lump sum prepayments by pairs of periods (or dates) and respective prepayment amounts.

You can specify the prepayment periods as dates if you specify the START= option. Prepayment periods or dates and the respective prepayment amounts must be in time sequence. The prepayments are treated as principal payments, and the outstanding principal balance is adjusted accordingly. In the adjustable rate and buydown rate loans, if there is a rate adjustment after prepayments, the adjusted periodic payment is calculated based on the outstanding principal balance. The prepayments do not result in periodic payment amount adjustments in fixed rate and balloon payment loans.

PRICE= *amount*
PRC= *amount*
specifies the purchase price, which is the loan amount plus the down payment. If you specify the PRICE= option along with the loan amount (AMOUNT= option) or the down payment (DOWNPAYMENT= option), the value of the other one is calculated.

If you specify the PRICE= option with the AMOUNTPCT= or DOWNPAYPCT= options, the loan amount and the downpayment are calculated.

ROUND= *n*
ROUND= NONE
specifies the number of decimal places to which the monetary amounts are rounded for the loan. Valid values for *n* are integers from 0 to 6. If you specify ROUND=NONE, the values are not rounded off internally, but the printed output is rounded off to two decimal places. The default is ROUND=2.

START= *SAS-date*
START= *yyyy:per*
S=
gives the date of loan initialization. The first payment is assumed to be one payment interval after the start date. For example, you can specify the START= option as '1APR1990'd or as 1990:3 where 3 is the third payment interval. If INTERVAL=QUARTER, 3 refers to the third quarter. If you specify the START= option, all output involving the particular loan is dated accordingly.

Output Options

NOSUMMARYPRINT
NOSUMPR
suppresses the printing of the loan summary report. The NOSUMMARYPRINT option is usually used when an OUTSUM= data set is created to store loan summary information.

NOPRINT
NOP
suppresses all printed output for the loan.

OUT= *SAS-data-set*
writes the loan amortization schedule to an output data set.

OUTSUM= *SAS-data-set*
writes the loan summary for the individual loan to an output data set.

SCHEDULE
SCHEDULE= *nyears*
SCHEDULE= YEARLY
SCHED
prints the amortization schedule for the loan. SCHEDULE=*nyears* specifies the number of years the printed amortization table covers. If you omit the number of years or specify a period longer than the loan life, the schedule is printed for the full term of the loan. SCHEDULE=YEARLY prints yearly summary information in the amortization schedule rather than the full amortization schedule. SCHEDULE=YEARLY is useful for long-term loans.

BALLOON Statement

BALLOON *options;*

The BALLOON statement specifies a fixed rate loan with scheduled balloon payments in addition to the periodic payment. The following option is used in the BALLOON statement, in addition to the required options listed under the FIXED statement;

BALLOONPAYMENT= *(date1=payment1 date2=payment2 ...)*
BALLOONPAYMENT= *(period1=payment1 period2=payment2 ...)*
BPAY=

specifies pairs of periods and amounts of balloon (lump sum) payments in excess of the periodic payment during the life of the loan. You can also specify the balloon periods as dates if you specify the START= option.

If you do not specify this option, the calculations are identical to a loan specified in a FIXED statement. Balloon periods (or dates) and the respective balloon payments must be in time sequence.

ARM Statement

ARM *options;*

The ARM statement specifies an adjustable rate loan where the future interest rates are not known with certainty but will vary within specified limits according to the terms stated in the loan agreement. In practice, the adjustment terms vary. Adjustments in the interest rate can be captured using the ARM statement options.

In addition to the required specifications and options listed under the FIXED statement, you can use the following options with the ARM statement:

Rate Adjustment Terms Options

ADJUSTFREQ= *n*
ADF= *n*

specifies the number of periods, in terms of the INTERVAL= specification, between rate adjustments. INTERVAL=MONTH ADJUSTFREQ=6 indicates that the nominal interest rate can be adjusted every six months until the life cap or maximum rate (whichever is specified) is reached. The default is ADJUSTFREQ=12. The periodic payment is adjusted every adjustment period even if there is no rate change; therefore, if prepayments are made (as specified with the PREPAYMENTS= option), the periodic payment might change even if the nominal rate does not.

CAPS= *(periodic-cap, life-cap* **)**

specifies the maximum interest rate adjustment, in percent notation, allowed by the loan agreement. The *periodic cap* specifies the maximum adjustment allowed at each adjustment period. The *life cap* specifies the maximum total adjustment over the life of the loan. For example, a loan specified with CAPS=(0.5, 2) indicates that the nominal interest rate can change by 0.5% each adjustment period, and the annual nominal interest rate throughout the life of the loan will be within a 2% range of the initial annual nominal rate.

MAXADJUST= *rate*
MAXAD= *rate*

specifies the maximum rate adjustment, in percent notation, allowed at each adjustment period. Use the MAXADJUST= option with the MAXRATE= and MINRATE= options. The initial nominal rate plus the maximum adjustment should not exceed the specified MAXRATE= value. The initial nominal rate minus the maximum adjustment should not be less than the specified MINRATE= value.

MAXRATE= *rate*
MAXR= *rate*

specifies the maximum annual nominal rate, in percent notation, that might be charged on the loan. The maximum annual nominal rate should be greater than or equal to the initial annual nominal rate specified with the RATE= option.

MINRATE= *rate*
MINR= *rate*

specifies the minimum annual nominal rate, in percent notation, that might be charged on the loan. The minimum annual nominal rate should be less than or equal to the initial annual nominal rate specified with the RATE= option.

Rate Adjustment Case Options

PROC LOAN supports four rate adjustment scenarios for analysis of adjustable rate loans: pessimistic (WORSTCASE), optimistic (BESTCASE), no-change (FIXEDCASE), and estimated (ESTIMATEDCASE). The estimated case enables you to analyze the adjustable rate loan using your predictions of future interest rates. The default is worst-case analysis. If more than one case is specified, worst-case analysis is performed. You can specify options for adjustable rate loans as follows:

BESTCASE
B

specifies a best-case analysis. The best-case analysis assumes the interest rate charged on the loan will reach its minimum allowed limits at each adjustment period and over the life of the loan. If you use the BESTCASE option, you must specify either the CAPS= option or the MINRATE= and MAXADJUST= options.

ESTIMATEDCASE= *(date1=rate1 date2=rate2 ...)*
ESTIMATEDCASE= *(period1=rate1 period2=rate2 ...)*
ESTC=

specifies an estimated case analysis that indicates the rate adjustments will follow the rates you predict. This option specifies pairs of periods and estimated nominal interest rates.

The ESTIMATEDCASE= option can specify adjustments that cannot fit into the BESTCASE, WORSTCASE, or FIXEDCASE specifications, or “what-if” type analysis. If you specify the START= option, you can also specify the estimation periods as dates. Estimated rates and the respective periods must be in time sequence.

If the estimated period falls between two adjustment periods (determined by ADJUSTFREQ= option), the rate is adjusted in the next adjustment period. The nominal interest rate charged on the loan is constant between two adjustment periods.

If any of the MAXRATE=, MINRATE=, CAPS=, and MAXADJUST= options are specified to indicate the rate adjustment terms of the loan agreement, these specifications are used to bound the rate adjustments. By using the ESTIMATEDCASE= option, you are predicting what the annual nominal rates in the market will be at different points in time, not necessarily the interest rate on your particular loan. For example, if the initial nominal rate (RATE= option) is 6.0, ADJUSTFREQ=6, MAXADJUST=0.5, and the ESTIMATEDCASE=(6=6.5, 12=7.5), the actual nominal rates charged on the loan would be 6.0% initially, 6.5% for the sixth through the eleventh periods, and 7.5% for the twelfth period onward.

FIXEDCASE
FIXCASE

specifies a fixed case analysis that assumes the rate will stay constant. The FIXEDCASE option calculates the ARM loan values similar to a fixed rate loan, but the payments are updated every adjustment period even if the rate does not change, leading to minor differences between the two methods. One such difference is in the way prepayments are handled. In a fixed rate loan, the rate and the payments are never adjusted; therefore, the payment stays the same over the life of the loan even when prepayments are made (instead, the life of the loan is shortened). In an ARM loan with the FIXEDCASE option, on the other hand, if prepayments are made, the payment is adjusted in the following adjustment period, leaving the life of the loan constant.

WORSTCASE
W

specifies a worst-case analysis. The worst-case analysis assumes the interest rate charged on the loan will reach its maximum allowed limits at each rate adjustment period and over the life of the loan. If the WORSTCASE option is used, either the CAPS= option or the MAXRATE= and MAXADJUST= options must be specified.

BUYDOWN Statement

BUYDOWN *options;*

The BUYDOWN statement specifies a buydown rate loan. The buydown rate loans are similar to ARM loans, but the interest rate adjustments are predetermined at the initialization of the loan, usually by paying interest points at the time of loan initialization.

You can use all the required specifications and options listed under the FIXED statement with the BUYDOWN statement. The following option is specific to the BUYDOWN statement and is required:

BUYDOWNRATES= *(date1=rate1 date2=rate2 ...)*
BUYDOWNRATES= *(period1=rate1 period2=rate2 ...)*
BDR=

specifies pairs of periods and the predetermined nominal interest rates that will be charged on the loan starting at the corresponding time periods.

You can also specify the buydown periods as dates if you specify the START=option. Buydown periods (or dates) and the respective buydown rates must be in time sequence.

COMPARE Statement

COMPARE *options ;*

The COMPARE statement compares multiple loans and it can be used with a single loan. You can use only one COMPARE statement. COMPARE statement options specify the periods and desired types of analysis for loan comparison. The default analysis reports the outstanding principal balance, breakeven of payment, breakeven of interest paid, and before-tax true interest rate. The default comparison period corresponds to the first LIFE= option specification. If the LIFE= option is not specified for any loan, the loan comparison period defaults to the first calculated life.

You can use the following options with the COMPARE statement. For more detailed information on loan comparison, see the section "Loan Comparison Details" later in this chapter.

Analysis Options

ALL
is equivalent to specifying the BREAKINTEREST, BREAKPAYMENT, PWOFCOST, and TRUEINTEREST options. The loan comparison report includes all the criteria. You need to specify the MARR= option for present worth of cost calculation.

AT= *(date1 date2 ...)*
AT= *(period1 period2 ...)*
specifies the periods for loan comparison reports. If you specify the START= option in the PROC LOAN statement, you can specify the AT= option as a list of dates instead of periods. The comparison periods do not need to be in time sequence. If you do not specify the AT= option, the comparison period defaults to the first LIFE= option specification. If you do not specify the LIFE= option for any of the loans, the loan comparison period defaults to the first calculated life.

BREAKINTEREST
BI
specifies breakeven analysis of the interest paid. The loan comparison report includes the interest paid for each loan through the specified comparison period (AT= option).

BREAKPAYMENT
BP
specifies breakeven analysis of payment. The periodic payment for each loan is reported for every comparison period specified in the AT=option.

MARR= *rate*
specifies the MARR (**M**inimum **A**ttractive **R**ate of **R**eturn) in percent notation. MARR reflects the cost of capital or the opportunity cost of money. The MARR= option is used in calculating the present worth of cost.

PWOFCOST
PWC

calculates the present worth of cost (net present value of costs) for each loan based on the cash flow through the specified comparison periods. The calculations account for down payment, initialization costs, and discount points, as well as the payments and outstanding principal balance at the comparison period. If you specify the TAXRATE= option, the present worth of cost is based on after-tax cash flow. Otherwise, before-tax present worth of cost is calculated. You need to specify the MARR= option for present worth of cost calculations.

TAXRATE= *rate*
TAX= *rate*

specifies income tax rate in percent notation for the after-tax calculations of the true interest rate and present worth of cost for those assets that qualify for tax deduction. If you specify this option, the amount specified in the POINTS= option and the interest paid on the loan are assumed to be tax-deductible. Otherwise, it is assumed that the asset does not qualify for tax deductions, and the cash flow is not adjusted for tax savings.

TRUEINTEREST
TI

calculates the true interest rate (effective interest rate based on the cash flow of all payments, initialization costs, discount points, and the outstanding principal balance at the comparison period) for all the specified loans through each comparison period. If you specify the TAXRATE= option, the true interest rate is based on after-tax cash flow. Otherwise, the before-tax true interest rate is calculated.

Output Options

NOCOMPRINT
NOCP

suppresses the printing of the loan comparison report. The NOCOMPRINT option is usually used when an OUTCOMP= data set is created to store loan comparison information.

OUTCOMP= *SAS-data-set*

writes the loan comparison report to an output data set.

Details

Computational Details

These terms are used in the formulas that follow:

p	periodic payment
a	principal amount
r_a	nominal annual rate
f	compounding frequency (per year)
f'	payment frequency (per year)
r	periodic rate
r_e	effective interest rate
n	total number of payments

The periodic rate, or the simple interest applied during a payment period, is given by

$$r = \left(1 + \frac{r_a}{f}\right)^{f/f'} - 1$$

Note that the interest calculation is performed at each payment period rather than at the compound period. This is done by adjusting the nominal rate. Refer to Muksian (1984) for details.

Note that when $f = f'$, that is, when the payment and compounding frequency coincide, the preceding expression reduces to the familiar form:

$$r = \frac{r_a}{f}$$

The periodic rate for continuous compounding can be obtained from this general expression by taking the limit as the compounding frequency *f* goes to infinity. The resulting expression is

$$r = \exp\left(\frac{\mathrm{r_a}}{\mathrm{f'}}\right) - 1$$

The effective interest rate, or annualized percentage rate (APR), is that rate which, if compounded once per year, is equivalent to the nominal annual rate compounded *f* times per year. Thus,

$$(1 + r_e) = (1 + r)^f = \left(1 + \frac{r_a}{f}\right)^f$$

or

$$r_e = \left(1 + \frac{r_a}{f}\right)^f - 1$$

For continuous compounding, the effective interest rate is given by

$$r_e = \exp\left(\mathrm{r_a}\right) - 1$$

Refer to Muksian (1984) for details.

The payment is calculated as

$$p = \frac{ar}{1 - \frac{1}{(1+r)^n}}$$

The amount is calculated as

$$a = \frac{p}{r}\left(1 - \frac{1}{(1+r)^n}\right)$$

Both the payment and amount are rounded to the nearest hundredth (cent) unless the ROUND= specification is different than the default, 2.

The total number of payments *n* is calculated as

$$n = \frac{-\ln\left(1 - \frac{ar}{p}\right)}{\ln(1+r)}$$

The total number of payments is rounded up to the nearest integer.

The nominal annual rate is calculated using the bisection method, with *a* as the objective and *r* starting in the interval between $8 * 10^{-6}$ and .1 with an initial midpoint .01 and successive midpoints bisecting.

Loan Comparison Details

In order to compare the costs of different alternatives, the input cash flow for the alternatives must be represented in equivalent values. The equivalent value of a cash flow accounts for the time-value of money. That is, it is preferable to pay the same amount of money later than to pay it now, since the money can earn interest while you keep it. The MARR (**M**inimum **A**ttractive **R**ate of **R**eturn) reflects the cost of capital or the opportunity cost of money, that is, the interest that would have been earned on the savings that is foregone by making the investment. The MARR is used to discount the cash flow of alternatives into equivalent values at a fixed point in time. The MARR can vary for each investor and for each investment. Therefore, the MARR= option must be specified in the COMPARE statement if present worth of cost (PWOFCOST option) comparison is specified.

Present worth of cost reflects the equivalent amount at loan initialization of the loan cash flow discounted at MARR, not accounting for inflation. Present worth of cost accounts for the down payment, initialization costs, discount points, periodic payments, and the principal balance at the end of the report period. Therefore, it reflects the present worth of cost of the asset, not the loan. It is only meaningful to use minimization of present worth of cost as a selection criterion if the assets (down payment plus loan amount) are of the same value.

Another economic selection criterion is the rate of return (internal rate of return) of the alternatives. If interest is being earned by an alternative, the objective would be to maximize the rate of return. If interest is being paid, as in loan alternatives, the best alternative is the one that minimizes the rate of return. The true interest rate reflects the effective annual rate charged on the loan based on the cash flow, including the initialization cost and the discount points.

The effects of taxes on different alternatives must be accounted for when these vary among different alternatives. Since interest costs on certain loans are tax-deductible, the comparisons for those loans are made based on the after-tax cash flows. The cost of the loan is reduced by the tax benefits it offers through the loan life if the TAXRATE= option is specified. The present worth of cost and true interest rate are calculated based on the after-tax cash flow of the loan. The down payment on the loan and initialization costs are assumed to be not tax-deductible in after-tax analysis. Discount points and the interest paid in each periodic payment are assumed to be tax-deductible if the TAXRATE= option is specified. If the TAXRATE= option is not specified, the present worth of cost and the true interest rate are based on before-tax cash flow, assuming that the interest paid on the specified loan does not qualify for tax benefits.

The other two selection criteria are breakeven analysis of periodic payment and interest paid. If the objective is to minimize the periodic payment, the best alternative would be the one with the minimum periodic payment. If the objective is to minimize the interest paid on the principal, then the best alternative is the one with the least interest paid.

Another criterion might be the minimization of the outstanding balance of the loan at a particular point in time. For example, if you plan to sell a house before the end of the loan life (which is often the case), you might want to select the loan with the minimum principal balance at the time of the sale, since this balance must be paid at that time. The outstanding balance of the alternative loans is calculated for each loan comparison period by default.

If you specified the START= option in the PROC LOAN statement, the present worth of cost reflects the equivalent amount for each loan at that point in time. Any loan that has a START= specification different from the one in the PROC LOAN statement is not processed in the loan comparison.

The loan comparison report for each comparison period contains for each loan the loan label, outstanding balance, and any of the following measures if requested in the COMPARE statement: periodic payment (BREAKPAYMENT option), total interest paid to date (BREAKINTEREST option), present worth of cost (PWOFCOST option), and true interest rate (TRUEINTEREST option). The best loan is selected on the basis of present worth of cost or true interest rate. If both PWOFCOST and TRUEINTEREST options are specified, present worth of cost is the basis for the selection of the best loan.

You can use the OUTCOMP= option in the COMPARE statement to write the loan comparison report to a data set. The NOCOMPRINT option suppresses the printing of a loan comparison report.

OUT= Data Set

The OUT= option writes the loan amortization schedule to an output data set. The OUT= data set contains one observation for each payment period (or one observation for each year if you specified the SCHEDULE=YEARLY option). If you specified the START= option, the DATE variable denotes the date of the payment. Otherwise, YEAR and period variable (SEMIMONTH, MONTH, QUARTER, or SEMIYEAR) denote the payment year and period within the year.

The OUT= data set contains the following variables:

- DATE, date of the payment. DATE is included in the OUT= data set only when you specify the START= option.
- YEAR, year of the payment period. YEAR is included in the OUT= data set only when you do not specify the START= option.
- PERIOD, period within the year of the payment period. The name of the period variable matches the INTERVAL= specification (SEMIMONTH, MONTH, QUARTER, or SEMIYEAR.) The PERIOD variable is included in the OUT= data set only when you do not specify the START= option.
- BEGPRIN, beginning principal balance
- PAYMENT, payment
- INTEREST, interest payment
- PRIN, principal repayment
- ENDPRIN, ending principal balance

OUTCOMP= Data Set

The OUTCOMP= option in the COMPARE statement writes the loan comparison analysis results to an output data set. If you specified the START= option, the DATE variable identifies the date of the loan comparison. Otherwise, the PERIOD variable identifies the comparison period.

The OUTCOMP= data set contains one observation for each loan for each loan comparison period. The OUTCOMP= data set contains the following variables.

- DATE, date of loan comparison report. The DATE variable is included in the OUTCOMP= data set only when you specify the START= option.
- PERIOD, period of the loan comparison for the observation. The PERIOD variable is included in the OUTCOMP= data set only when you do not specify the START= option.
- LABEL, label string for the loan
- TYPE, type of the loan
- PAYMENT, periodic payment at the time of report. The PAYMENT is included in the OUTCOMP= data set if you specified the BREAKPAYMENT or ALL option or if you used default criteria.
- INTPAY, interest paid through the time of report. The INTPAY variable is included in the OUTCOMP= data set if you specified the BREAKINTEREST or ALL option or if you used default criteria.
- TRUERATE, true interest rate charged on the loan. The TRUERATE variable is included in the OUTCOMP= data set if you specified the TRUERATE or ALL option or if you used default criteria.
- PWOFCOST, present worth of cost. The PWOFCOST variable is included in the OUTCOMP= data set only if you specified the PWOFCOST or ALL option.
- BALANCE, outstanding principal balance at the time of report

OUTSUM= Data Set

The OUTSUM= option writes the loan summary to an output data set. If you specified this option in the PROC LOAN statement, the loan summary information for all loans will be written to the specified data set, except for those loans for which you specified a different OUTSUM= data set on the ARM, BALLOON, BUYDOWN, or FIXED statement.

The OUTSUM= data set contains one observation for each loan and contains the following variables:

- TYPE, type of loan
- LABEL, loan label
- PAYMENT, periodic payment
- AMOUNT, loan principal
- DOWNPAY, down payment. DOWNPAY is included in the OUTSUM= data set only when you specify a down payment.
- INITIAL, loan initialization costs. INITIAL is included in the OUTSUM= data set only when you specify initialization costs.
- POINTS, discount points. POINTS is included in the OUTSUM= data set only when you specify discount points.
- TOTAL, total payment

- INTEREST, total interest paid
- RATE, nominal annual interest rate
- EFFRATE, effective interest rate
- INTERVAL, payment interval
- COMPOUND, compounding interval
- LIFE, loan life (that is, the number of payment intervals)
- NCOMPND, number of compounding intervals
- COMPUTE, computed loan parameter: life, amount, payment, or rate

If you specified the START= option either in the PROC LOAN statement or for the individual loan, the OUTSUM= data set also contains the following variables:

- BEGIN, start date
- END, loan termination date

Printed Output

The output from PROC LOAN consists of the loan summary table, loan amortization schedule, and loan comparison report.

Loan Summary Table

The loan summary table shows the total payment and interest, the initial nominal annual and effective interest rates, payment and compounding intervals, the length of the loan in the time units specified, the start and end dates if specified, a list of nominal and effective interest rates, and periodic payments throughout the life of the loan.

A list of balloon payments for balloon payment loans and a list of prepayments if specified are printed with their respective periods or dates.

The loan summary table is printed for each loan by default. The NOSUMMARYPRINT option specified in the PROC LOAN statement will suppress the printing of the loan summary table for all loans. The NOSUMMARYPRINT option can be specified in individual loan statements to selectively suppress the printing of the loan summary table.

Loan Repayment Schedule

The amortization schedule contains for each payment period, the year and period within the year (or date, if you specified the START= option), principal balance at the beginning of the period, total payment, interest payment and principal payment for the period, and the principal balance at the end of the period. If you specified the SCHEDULE=YEARLY option, the amortization will contain a summary for each year instead of for each payment period.

The amortization schedule is not printed by default. The SCHEDULE option in the PROC LOAN statement requests the printing of amortization tables for all loans. You can specify the SCHEDULE option in individual loan statements to selectively request the printing of the amortization schedule.

Loan Comparison Report

The loan comparison report is processed for each report period and contains the results of economic analysis of the loans. The quantities reported can include the outstanding principal balance, after-tax or before-tax present worth of cost and true interest rate, periodic payment, and the interest paid through the report period for each loan. The best alternative is identified if the asset value (down payment plus loan amount) is the same for each alternative.

The loan comparison report is printed by default. The NOCOMPRINT option specified in the COMPARE statement suppresses the printing of the loan comparison report.

ODS Table Names

PROC LOAN assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 18.1. ODS Tables Produced in PROC LOAN

ODS Table Name	Description	Option
ODS Tables Created by the PROC LOAN, FIXED, ARM, BALLOON, and BUYDOWN Statements		
Repayment	Loan Repayment Schedule	SCHEDULE
ODS Tables Created by the FIXED, ARM, BALLOON, and BUYDOWN Statements		
LoanSummary	Loan Summary	default
RateList	Rates and Payments	default
PrepayList	Prepayments and Periods	PREPAYMENTS=
ODS Tables Created by the BALLOON Statement		
BalloonList	Balloon Payments and Periods	default
ODS Tables Created by the COMPARE Statement		
Comparison	Loan Comparison Report	default

Examples

Example 18.1. Discount Points for Lower Interest Rates

This example illustrates the comparison of two $100,000 loans. The major difference between the two loans is that the nominal interest rate in the second one is lower than the first with the added expense of paying discount points at the time of initialization.

Both alternatives are 30-year loans. The first loan is labeled “8.25% - no discount points” and the second one is labeled “8% - 1 discount point.”

Assume that the interest paid qualifies for a tax deduction, and you are in the 33% tax bracket. Also, your Minimum Attractive Rate of Return (MARR) for an alternative investment is 4% (adjusted for tax rate.)

You use the following statements to find the breakeven point in the life of the loan for your preference between the loans:

```
proc loan start=1992:1 nosummaryprint amount=100000 life=360;
   fixed rate=8.25 label='8.25% - no discount points';
   fixed rate=8 points=1000 label='8% - 1 discount point';
   compare at=(48 54 60) all taxrate=33 marr=4;
run;
```

Output 18.1.1 shows the loan comparison reports as of January 1996 (48th period), July 1996 (54th period), and January 1997 (60th period).

Output 18.1.1. Loan Comparison Reports for Discount Point Breakeven

```
                                       The LOAN Procedure

                                     Loan Comparison Report

                                   Analysis through JAN1996

                                     Ending    Present Worth                         Interest       True
Loan Label                      Outstanding          of Cost          Payment            Paid       Rate

8.25% - no discount                96388.09        105546.17           751.27        32449.05       5.67
points
8% - 1 discount point              96219.32        105604.05           733.76        31439.80       5.69

NOTE: "8.25 - no discount points" is the best alternative based on present worth of cost analysis
                                         through JAN1996.


                                     Loan Comparison Report

                                   Analysis through JUL1996

                                     Ending    Present Worth                         Interest       True
Loan Label                      Outstanding          of Cost          Payment            Paid       Rate

8.25% - no discount                95847.27        106164.97           751.27        36415.85       5.67
points
8% - 1 discount point              95656.22        106153.97           733.76        35279.26       5.67

  NOTE: "8 - 1 discount point" is the best alternative based on present worth of cost analysis
                                         through JUL1996.
```

```
                                    The LOAN Procedure

                                  Loan Comparison Report

                                Analysis through JAN1997

                                    Ending    Present Worth                    Interest      True
Loan Label                     Outstanding          of Cost        Payment         Paid      Rate

8.25% - no discount               95283.74        106768.07         751.27     40359.94      5.67
points
8% - 1 discount point             95070.21        106689.80         733.76     39095.81      5.66

  NOTE: "8 - 1 discount point" is the best alternative based on present worth of cost analysis
                                     through JAN1997.
```

Notice that the breakeven point for present worth of cost and true rate both happen on July 1996. This indicates that if you intend to keep the loan for 4.5 years or more, it is better to pay the discount points for the lower rate. If your objective is to minimize the interest paid or the periodic payment, the “8% - 1 discount point” loan is the preferred choice.

Example 18.2. Refinancing a Loan

Assume that you obtained a fixed rate 15-year loan in June 1995 for $78,500 with a nominal annual rate of 9%. By early 1998, the market offers a 6.5% interest rate, and you are considering whether to refinance your loan.

Use the following statements to find out the status of the loan on February 1998. Output 18.2.1 shows the results:

```
proc loan start=1995:6;
   fixed life=180 rate=9 amount=78500 noprint
         label='Original Loan';
   compare at=('10FEB1998'd);
run;
```

Output 18.2.1. Loan Comparison Report for Original Loan

```
                                    The LOAN Procedure

                                  Loan Comparison Report

                                Analysis through FEB1998

                                       Ending                      Interest       True
     Loan Label                   Outstanding         Payment          Paid       Rate

     Original Loan                   71028.75          796.20      18007.15       9.38
```

The monthly payment on the original loan is $796.20. The ending outstanding principal balance as of February is $71,028.75. At this point, you might want to refinance your loan with another 15-year loan. The alternate loan has a 6.5% nominal annual rate. The initialization costs are $1,419.00. Use the following statements to compare your alternatives:

```
proc loan start=1998:2 amount=71028.75;
   fixed rate=9 payment=796.20
```

```
         label='Keep the original loan' noprint;
   fixed life=180 rate=6.5 init=1419
         label='Refinance at 6.5%' noprint;
   compare at=(15 16) taxrate=33 marr=4 all;
run;
```

The comparison reports of May 1999 and June 1999 in Output 18.2.2 illustrate the breakeven between the two alternatives. If you intend to keep the loan through June 1999 or longer, your initialization costs for the refinancing are justified. The periodic payment of the refinanced loan is $618.74.

Output 18.2.2. Loan Comparison Report for Refinancing Decision

```
                                            The LOAN Procedure

                                          Loan Comparison Report

                                        Analysis through MAY1999

                                Ending    Present Worth                         Interest       True
Loan Label                 Outstanding          of Cost        Payment             Paid        Rate

Keep the original loan        66862.10         72737.27         796.20          7776.35        6.20
Refinance at 6.5%             67382.48         72747.51         618.74          5634.83        6.23

 NOTE: "Keep the original loan" is the best alternative based on present worth of cost analysis
                                         through MAY1999.


                                          Loan Comparison Report

                                        Analysis through JUN1999

                                Ending    Present Worth                         Interest       True
Loan Label                 Outstanding          of Cost        Payment             Paid        Rate

Keep the original loan        66567.37         72844.52         796.20          8277.82        6.20
Refinance at 6.5%             67128.73         72766.42         618.74          5999.82        6.12

NOTE: "Refinance at 6.5" is the best alternative based on present worth of cost analysis through
                                             JUN1999.
```

Example 18.3. Prepayments on a Loan

This example compares a 30-year loan with and without prepayments. Assume the 30-year loan has an 8.25% nominal annual rate. Use the following statements to see the effect of making uniform prepayments of $500 with periodic payment:

```
proc loan start=1992:12 rate=8.25 amount=240000 life=360;
   fixed label='No prepayments';
   fixed label='With Prepayments' prepay=500 ;
   compare at=(120) taxrate=33 marr=4 all;
run;
```

Output 18.3.1. Loan Summary Reports and Loan Comparison Report

```
                              The LOAN Procedure

                           Fixed Rate Loan Summary

                                No prepayments

   Downpayment                 0.00    Principal Amount          240000.00
   Initialization              0.00    Points                         0.00
   Total Interest         409094.17    Nominal Rate                8.2500%
   Total Payment          649094.17    Effective Rate              8.5692%
   Pay Interval             MONTHLY    Compounding                 MONTHLY
   No. of Payments              360    No. of Compoundings             360
   Start Date               DEC1992    End Date                    DEC2022


                    Rates and Payments for No prepayments
        Date          Nominal Rate     Effective Rate         Payment

        DEC1992           8.2500%           8.5692%           1803.04
```

```
                              The LOAN Procedure

                           Fixed Rate Loan Summary

                               With Prepayments

   Downpayment                 0.00    Principal Amount          240000.00
   Initialization              0.00    Points                         0.00
   Total Interest         183650.70    Nominal Rate                8.2500%
   Total Payment          423650.70    Effective Rate              8.5692%
   Pay Interval             MONTHLY    Compounding                 MONTHLY
   No. of Payments              184    No. of Compoundings             184
   Start Date               DEC1992    End Date                    APR2008


                   Rates and Payments for With Prepayments
        Date          Nominal Rate     Effective Rate         Payment

        DEC1992           8.2500%           8.5692%           2303.04
```

```
                              The LOAN Procedure

                            Loan Comparison Report

                           Analysis through DEC2002

                              Ending    Present Worth                  Interest        True
Loan Label               Outstanding          of Cost        Payment       Paid        Rate

No prepayments             211608.05        268762.31        1803.04  187972.85        5.67
With Prepayments           118848.23        264149.25        2303.04  155213.03        5.67

NOTE: "With Prepayments" is the best alternative based on present worth of cost analysis through
                                   DEC2002.
```

Output 18.3.1 illustrates the Loan Summary Reports and the Loan Comparison report. Notice that with prepayments you pay off the loan in slightly more than 15 years. Also, the total payments and total interest are considerably lower with the prepayments. If you can afford the prepayments of $500 each month, another alternative you should consider is using a 15-year loan, which is generally offered at a lower nominal interest rate.

Example 18.4. Output Data Sets

This example shows the analysis and comparison of five alternative loans. Initialization cost, discount points, and both lump sum and periodic payments are included in the specification of these loans. Although no printed output is produced, the loan summary and loan comparison information is stored in the OUTSUM= and OUTCOMP= data sets.

```
proc loan start=1998:12 noprint outsum=loans
   amount=150000 life=360;

   fixed rate=7.5 life=180 prepayment=500
         label='BANK1, Fixed Rate';

   arm rate=5.5 estimatedcase=(12=7.5 18=8)
       label='BANK1, Adjustable Rate';

   buydown rate=7 interval=semimonth init=15000
           bdrates=(3=9 10=10) label='BANK2, Buydown';

   arm rate=5.75 worstcase caps=(0.5 2.5)
       adjustfreq=6 label='BANK3, Adjustable Rate'
       prepayments=(12=2000 36=5000);

   balloon rate=7.5 life=480
       points=1100 balloonpayment=(15=2000 48=2000)
       label='BANK4, with Balloon Payment';

   compare at=(120 360) all marr=7 tax=33 outcomp=comp;
run;

proc print data=loans;
run;

proc print data=comp;
run;
```

Output 18.4.1 illustrates the contents of the output data sets.

Output 18.4.1. OUTSUM= and OUTCOMP= Data Sets

```
Obs  TYPE     LABEL                           PAYMENT  AMOUNT  INITIAL  POINTS    TOTAL     INTEREST

 1   FIXED    BANK1, Fixed Rate               1890.52  150000       0       0   207839.44   57839.44
 2   ARM      BANK1, Adjustable Rate           851.68  150000       0       0   390325.49  240325.49
 3   BUYDOWN  BANK2, Buydown                   673.57  150000   15000       0   288858.08  138858.08
 4   ARM      BANK3, Adjustable Rate           875.36  150000       0       0   387647.82  237647.82
 5   BALLOON  BANK4, with Balloon Payment      965.36  150000       0    1100   467372.31  317372.31

Obs   RATE     EFFRATE   INTERVAL      COMPOUND      LIFE   NCOMPND   COMPUTE     START        END

 1   0.0750   0.077633   MONTHLY       MONTHLY        110     110     PAYMENT   DEC1998   FEB2008
 2   0.0550   0.056408   MONTHLY       MONTHLY        360     360     PAYMENT   DEC1998   DEC2028
 3   0.0700   0.072399   SEMIMONTHLY   SEMIMONTHLY    360     360     PAYMENT   DEC1998   DEC2013
 4   0.0575   0.059040   MONTHLY       MONTHLY        360     360     PAYMENT   DEC1998   DEC2028
 5   0.0750   0.077633   MONTHLY       MONTHLY        480     480     PAYMENT   DEC1998   DEC2038
 Obs     DATE TYPE     LABEL                         PAYMENT   INTEREST TRUERATE   PWOFCOST    BALANCE

   1 DEC2008 FIXED   BANK1, Fixed Rate            1772.76  57839.44 0.051424 137741.07      0.00
   2 DEC2008 ARM     BANK1, Adjustable Rate       1093.97 108561.77 0.052212 130397.88 130788.65
   3 DEC2008 BUYDOWN BANK2, Buydown                803.98 118182.19 0.087784 161810.00  75798.19
   4 DEC2008 ARM     BANK3, Adjustable Rate       1065.18 107015.58 0.053231 131955.90 125011.88
   5 DEC2008 BALLOON BANK4, with Balloon Payment   965.36 107906.61 0.052107 130242.56 138063.41
   6 DEC2028 FIXED   BANK1, Fixed Rate            1772.76  57839.44 0.051424 137741.07      0.00
   7 DEC2028 ARM     BANK1, Adjustable Rate       1094.01 240325.49 0.053247 121980.94      0.00
   8 DEC2028 BUYDOWN BANK2, Buydown                800.46 138858.08 0.086079 161536.44      0.00
   9 DEC2028 ARM     BANK3, Adjustable Rate       1065.20 237647.82 0.054528 124700.22      0.00
  10 DEC2028 BALLOON BANK4, with Balloon Payment   965.36 282855.86 0.051800 117294.50  81326.26
```

Example 18.5. Piggyback Loans

The *piggyback* loan is becoming a widely available alternative. Borrowers would like to avoid the PMI (Private Mortgage Insurance) required with loans where the borrower has a downpayment of less than 20% of the price. The piggyback allows a secondary home equity loan to be packaged with a primary loan with less than 20% downpayment. The secondary loan usually has a shorter life and higher interest rate. The interest paid on both loans are tax-deductible whereas PMI does not qualify for a tax deduction.

The following example compares a conventional fixed rate loan with 20% down as opposed to a piggyback loan: one primary fixed rate with 10% downpayment and a secondary, home equity loan for 10% of the original price. All loans have monthly payments.

The conventional loan alternative is a 30 year loan with a fixed annual rate of 7.5%. The primary loan in the piggyback loan setup is also a 30 year loan with a fixed annual rate of 7.75%. The secondary loan is a 15 year loan with a fixed annual interest rate of 8.25%.

The comparison output for the two loans comprising the piggyback loan is aggregated using the TIMESERIES procedure with a minimum of specified options:

- The INTERVAL= option requests that the data be aggregated into periods of length 5 years beginning on the 25th month, resulting in appropriately date identified periods.
- The ACC=TOTAL option specifies that the output should reflect accumulated totals as opposed to, say, averages.
- The NOTSORTED option indicates that the input data set has not been sorted by the ID variable.

See Chapter 28, “The TIMESERIES Procedure,” for more information on this procedure.

Use the following statements to analyze the conventional loan, as well as the piggyback alternative, and compare them on the basis of their present value of cost, outstanding balance, and interest payment amounts at the end of 5, 10, and 15 years into the loan life.

```
TITLE1 'LOAN: Piggyback loan example';

TITLE2 'LOAN: Conventional loan';

proc loan start=2002:1 noprint;

     fixed price=200000 dp=40000 rate=7.5 life=360
           label='20 percent down: Conventional Fixed Rate' ;

     compare at=(60 120 180) pwofcost taxrate=30 marr=12
            breakpay breakint outcomp=comploans;

run;

TITLE2 'LOAN: Piggyback: Primary Loan';

proc loan start=2002:1 noprint;

     fixed amount=160000 dp=20000 rate=7.75 life=360
           label='Piggyback: Primary loan' out=loan1;

     compare at=(60 120 180 ) pwofcost taxrate=30 marr=12
            breakpay breakint outcomp=cloan1;

run;

TITLE2 'LOAN: Piggyback: Secondary (Home Equity) Loan';

proc loan start=2002:1 noprint;

     fixed amount=20000  rate=8.25 life=180
           label='Piggyback: Secondary (Home Equity) Loan' out=loan2;

     compare at=(60 120 180 ) pwofcost taxrate=30 marr=12
            breakpay breakint  outcomp=cloan2;

run;

data cloan12;

set cloan1  cloan2;

run;


proc timeseries data=cloan12 out= totcomp ;
id date interval=year5.25 acc=total notsorted;
var payment interest pwofcost balance ;
run;

TITLE2 'LOAN: Piggyback loan';
proc print data=totcomp;
format date monyy7.;
run;
```

```
data comploans;
set comploans;
drop type label;
run;


TITLE2 'LOAN: Conventional Loan';

proc print data=comploans; run;
```

The loan comparisons in Output 18.5.1 illustrate the aftertax comparison of the loans. The after tax present value of cost for the piggyback loan is lower than the 20% down conventional fixed rate loan.

Output 18.5.1. Piggyback Loan

```
                          LOAN: Piggyback loan example
                              LOAN: Piggyback loan

              Obs       DATE     PAYMENT     INTEREST     PWOFCOST     BALANCE

               1      JAN2007    1285.51     63955.94    155188.50    166825.34
               2      JAN2012    1285.51    121870.62    132685.24    147609.42
               3      JAN2017    1284.88    170958.63    122033.55    119567.46



                          LOAN: Piggyback loan example
                            LOAN: Conventional Loan

              Obs       DATE     PAYMENT     INTEREST     PWOFCOST     BALANCE

               1      JAN2007    1174.02     62553.27    162413.00    152112.07
               2      JAN2012    1174.02    121242.63    142973.12    140360.23
               3      JAN2017    1174.02    174175.43    133297.46    122851.83
```

References

DeGarmo, E.P., Sullivan, W.G., and Canada, J.R. (1984), *Engineering Economy*, Seventh Edition, New York: Macmillan Publishing Company.

Muksian, R. (1984), *Financial Mathematics Handbook*, Englewood Cliffs, NJ: Prentice-Hall.

Newnan, D.G. (1988), *Engineering Economic Analysis*, Third Edition, San Jose, CA: Engineering Press.

Riggs, J.L. and West, T.M. (1986), *Essentials of Engineering Economics*, Second Edition, New York: McGraw-Hill, Inc.

Chapter 19
The MDC Procedure

Chapter Contents

Chapter 19
The MDC Procedure

Overview

The MDC (Multinomial Discrete Choice) procedure analyzes models where the choice set consists of multiple alternatives. This procedure supports conditional logit, mixed logit, heteroscedastic extreme value, nested logit, and multinomial probit models. The MDC procedure uses the maximum likelihood (ML) or simulated maximum likelihood method for model estimation. The term *multinomial logit* is often used in the econometrics literature to refer to the *conditional logit* model of McFadden (1974). Here, the term *conditional logit* refers to McFadden's conditional logit model, and the term *multinomial logit* refers to a model that differs slightly. Schmidt and Strauss (1975) and Theil (1969) are early applications of the multinomial logit model in the econometrics literature. The main difference between McFadden's conditional logit model and the multinomial logit model is that the multinomial logit model makes the choice probabilities depend on the characteristics of the individuals only, whereas the conditional logit model considers the effects of choice attributes on choice probabilities as well.

Unordered multiple choices are observed in many settings in different areas of application. For example, choices of housing location, occupation, political party affiliation, type of automobile, and mode of transportation are all unordered multiple choices. Economics and psychology models often explain observed choices using the *random utility* function. The utility of a specific choice can be interpreted as the relative pleasure or happiness that the decision maker derives from that choice with respect to other alternatives in a finite choice set. It is assumed that the individual chooses the alternative for which the associated utility is highest. However, the utilities are not known to the analyst with certainty and are therefore treated by the analyst as random variables. When the utility function contains a random component, the individual choice behavior becomes a probabilistic process.

The random utility function of individual i for choice j can be decomposed into deterministic and stochastic components.

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where V_{ij} is a deterministic utility function, assumed to be linear in the explanatory variables, and ϵ_{ij} is an unobserved random variable that captures the factors that affect utility that are not included in V_{ij}. Different assumptions on the distribution of the errors, ϵ_{ij}, give rise to different classes of models.

The features of discrete choice models available in the MDC procedure are summarized in Table 19.1.

Table 19.1. Summary of Models Supported by PROC MDC

Model Type	**Utility Function**	**Distribution of ϵ_{ij}**
Conditional Logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	IEV independent and identical
HEV	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	HEV independent and nonidentical
Nested Logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	GEV correlated and identical
Mixed Logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij} + \epsilon_{ij}$	IEV independent and identical
Multinomial Probit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	MVN correlated and nonidentical

IEV stands for type I Extreme Value (or Gumbel) distribution with the probability density function and the cumulative distribution function of the random error given by $f(\epsilon_{ij}) = \exp(-\epsilon_{ij})\exp(-\exp(-\epsilon_{ij}))$ and $F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}))$; HEV stands for Heteroscedastic Extreme Value distribution with the probability density function and the cumulative distribution function of the random error given by $f(\epsilon_{ij}) = \frac{1}{\theta_j}\exp(\frac{\epsilon_{ij}}{\theta_j})\exp[-\exp(-\frac{\epsilon_{ij}}{\theta_j})]$ and $F(\epsilon_{ij}) = \exp[-\exp(-\frac{\epsilon_{ij}}{\theta_j})]$ where θ_j is a scale parameter for the random component of the jth alternative; GEV stands for Generalized Extreme Value distribution; MVN represents Multivariate Normal distribution; and ξ_{ij} is an error component. See the "Mixed Logit Model" section on page 961 for more information on ξ_{ij}.

Getting Started

Conditional Logit: Estimation and Prediction

The MDC procedure is similar in use to the other regression model procedures in the SAS System. However, the MDC procedure requires identification and choice variables. For example, consider a random utility function

$$U_{ij} = x_{1,ij}\beta_1 + x_{2,ij}\beta_2 + \epsilon_{ij} \quad j = 1, \ldots, 3$$

where the cumulative distribution function of the stochastic component is type I extreme value, $F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}))$. You can estimate this conditional logit model with the following statements:

```
proc mdc;
   model decision = x1 x2 / type=clogit
      choice=(mode 1 2 3);
   id pid;
run;
```

Note that the MDC procedure does not include the intercept term automatically like other regression procedures. The dependent variable decision takes the value 1 when a specific alternative is chosen; otherwise it takes the value 0. Each individual is allowed to choose one and only one of the possible alternatives. In other words, the variable decision takes the value 1 one time only for each individual. If each individual has three elements (1, 2, and 3) in the choice set, the NCHOICE=3 option can be specified instead of CHOICE=(mode 1 2 3).

Consider the following trinomial data from Daganzo (1979). The original data (origdata) contains travel time (ttime1-ttime3) and choice (choice) variables. ttime1-ttime3 are the travel times for three different modes of transportation, and choice indicates which one of the three modes is chosen. The choice variable must have integer values.

```
data origdata;
   input ttime1 ttime2 ttime3 choice @@;
   datalines;
16.481  16.196  23.89   2  15.123  11.373  14.182  2
19.469  8.822   20.819  2  18.847  15.649  21.28   2
12.578  10.671  18.335  2  11.513  20.582  27.838  1
10.651  15.537  17.418  1  8.359   15.675  21.05   1
11.679  12.668  23.104  1  23.237  10.356  21.346  2
13.236  16.019  10.087  3  20.052  16.861  14.168  3
18.917  14.764  21.564  2  18.2    6.868   19.095  2
10.777  16.554  15.938  1  20.003  6.377   9.314   2
19.768  8.523   18.96   2  8.151   13.845  17.643  2
22.173  18.045  15.535  1  13.134  11.067  19.108  2
14.051  14.247  15.764  1  14.685  10.811  12.361  3
11.666  10.758  16.445  1  17.211  15.201  17.059  3
13.93   16.227  22.024  1  15.237  14.345  19.984  2
10.84   11.071  10.188  1  16.841  11.224  13.417  2
13.913  16.991  26.618  3  13.089  9.822   19.162  2
16.626  10.725  15.285  3  13.477  15.509  24.421  2
20.851  14.557  19.8    2  11.365  12.673  22.212  2
13.296  10.076  17.81   2  15.417  14.103  21.05   1
15.938  11.18   19.851  2  19.034  14.125  19.764  2
10.466  12.841  18.54   1  15.799  16.979  13.074  3
12.713  15.105  13.629  2  16.908  10.958  19.713  2
17.098  6.853   14.502  2  18.608  14.286  18.301  2
11.059  10.812  20.121  1  15.641  10.754  24.669  2
7.822   18.949  16.904  1  12.824  5.697   19.183  2
11.852  12.147  15.672  2  15.557  8.307   22.286  2
;
```

A new data set (newdata) is created since PROC MDC requires that each individual decision maker has one case for each alternative in his choice set. Note that the ID statement is required for all MDC models. In the following example, there are two public transportation modes, 1 and 2, and one private transportation mode, 3, and all individuals share the same choice set.

```
data newdata(keep=pid decision mode ttime);
   set origdata;
   array tvec{3} ttime1 - ttime3;
   retain pid 0;
   pid + 1;
```

```
      do i = 1 to 3;
         mode = i;
         ttime = tvec{i};
         decision = ( choice = i );
         output;
      end;
   run;
```

The first nine observations of the transformed data set are as follows:

```
                    Obs     pid     mode      ttime     decision

                      1       1        1     16.481         0
                      2       1        2     16.196         1
                      3       1        3     23.890         0
                      4       2        1     15.123         0
                      5       2        2     11.373         1
                      6       2        3     14.182         0
                      7       3        1     19.469         0
                      8       3        2      8.822         1
                      9       3        3     20.819         0
```

Figure 19.1. Transformed Modal Choice Data

The decision variable, decision, must have one nonzero value for each decision maker corresponding to the actual choice. When the RANK option is specified, the decision variable may contain rank data. For more details, see the "MODEL Statement" section on page 946. The following SAS statements estimate the conditional logit model using maximum likelihood:

```
   proc mdc data=newdata;
      model decision = ttime / type=clogit nchoice=3
         optmethod=qn covest=hess;
      id pid;
   run;
```

When all individuals have the same choice set, the NCHOICE= option can be used instead of the CHOICE= option. However, the NCHOICE= option is not allowed when a nested logit model is estimated. When the NCHOICE=$number$ option is specified, the choices are generated as $1, \ldots, number$. For more flexible alternatives (e.g., 1 3 6 8), you need to use the CHOICE= option. The choice variable must have integer values.

The OPTMETHOD=QN option specifies the quasi-Newton optimization technique. The covariance matrix of the parameter estimates is obtained from the Hessian matrix since COVEST=HESS is specified. You may also specify COVEST=OP or COVEST=QML. See the "MODEL Statement" section on page 946 for more details.

The MDC procedure produces a summary of model estimation displayed in Figure 19.2. Since there are multiple observations for each individual, the "Number of Cases" (150) is larger than the number of individuals, "Number of Observations"

(50). Figure 19.3 shows the frequency distribution of the three choice alternatives. In this example, mode 2 is most frequently chosen.

```
                          The MDC Procedure

                     Conditional Logit Estimates

                          Model Fit Summary

                 Dependent Variable                       decision
                 Number of Observations                         50
                 Number of Cases                               150
                 Log Likelihood                          -33.32132
                 Maximum Absolute Gradient              2.97024E-6
                 Number of Iterations                            6
                 Optimization Method              Dual Quasi-Newton
                 AIC                                      68.64265
                 Schwarz Criterion                        70.55467
```

Figure 19.2. Estimation Summary Table

```
                          The MDC Procedure

                     Conditional Logit Estimates

                      Discrete Response Profile

                  Index    CHOICE     Frequency    Percent

                    0           1            14      28.00
                    1           2            29      58.00
                    2           3             7      14.00
```

Figure 19.3. Choice Frequency

The MDC procedure computes nine goodness-of-fit measures for the discrete choice model. Seven of them are pseudo-R^2 measures based on the null hypothesis that all coefficients except for an intercept term are zero (Figure 19.4). McFadden's likelihood ratio index (LRI) is the smallest in value.

```
                              The MDC Procedure

                         Conditional Logit Estimates

                           Goodness-of-Fit Measures

    Measure                         Value    Formula

    Likelihood Ratio (R)           43.219    2 * (LogL - LogL0)
    Upper Bound of R (U)           109.86    - 2 * LogL0
    Aldrich-Nelson                 0.4636    R / (R+N)
    Cragg-Uhler 1                  0.5787    1 - exp(-R/N)
    Cragg-Uhler 2                   0.651    (1-exp(-R/N)) / (1-exp(-U/N))
    Estrella                       0.6666    1 - (1-R/U)^(U/N)
    Adjusted Estrella              0.6442    1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
    McFadden's LRI                 0.3934    R / U
    Veall-Zimmermann               0.6746    (R * (U+N)) / (U * (R+N))

    N = # of observations, K = # of regressors
```

Figure 19.4. Likelihood Ratio Test and R^2 Measures

Finally, the parameter estimate is displayed in Figure 19.5.

```
                              The MDC Procedure

                         Conditional Logit Estimates

                             Parameter Estimates

                                             Standard                 Approx
        Parameter     DF     Estimate           Error    t Value    Pr > |t|

        ttime          1      -0.3572          0.0776      -4.60      <.0001
```

Figure 19.5. Parameter Estimate of Conditional Logit

The predicted choice probabilities are produced using the OUTPUT statement.

```
output out=probdata pred=p;
```

The parameter estimates can be used to forecast the choice probability of individuals that are not in the input data set. To do so, you need to append to the input data set extra observations whose values of the dependent variable decision are missing, since these extra observations are not supposed to be used in the estimation stage. The identification variable pid must have values that are not used in the existing observations. The output data set, probdata, contains a new variable, p, in addition to input variables in the data set, extdata.

```
data extra;
   input pid mode decision ttime;
   datalines;
51  1  .   5.0
51  2  .  15.0
51  3  .  14.0
;
data extdata;
   set newdata extra;
run;

proc mdc data=extdata;
   model decision = ttime /
         type=clogit covest=hess
         nchoice=3;
   id pid;
   output out=probdata pred=p;
run;

proc print data=probdata( where=( pid >= 49 ) );
   var mode decision p ttime;
   id pid;
run;
```

The last nine observations from the forecast data set (probdata) are displayed in Figure 19.6. It is expected that the decision maker will choose mode "1" based on predicted probabilities for all modes.

```
               pid     mode     decision        p         ttime

                49       1          0       0.46393     11.852
                49       2          1       0.41753     12.147
                49       3          0       0.11853     15.672
                50       1          0       0.06936     15.557
                50       2          1       0.92437      8.307
                50       3          0       0.00627     22.286
                51       1          .       0.93611      5.000
                51       2          .       0.02630     15.000
                51       3          .       0.03759     14.000
```

Figure 19.6. Out-Of-Sample Mode Choice Forecast

Nested Logit Modeling

A more general model can be specified using the nested logit model. Since the public transportation modes, 1 and 2, tend to be correlated, these two choices can be grouped together. The decision tree displayed in Figure 19.7 is constructed.

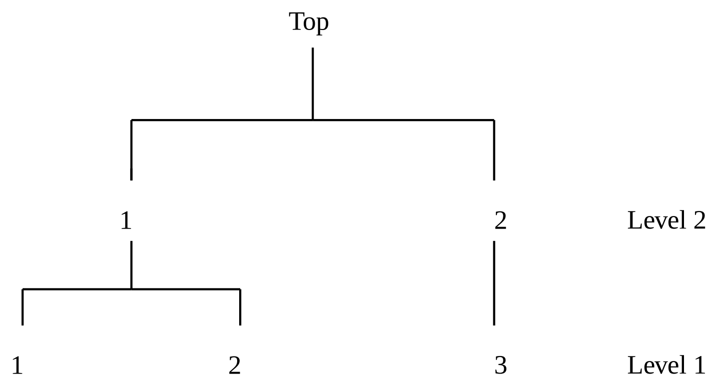


Figure 19.7. Decision Tree for Modal Choice

The two-level decision tree is specified in the NEST statement. The NCHOICE= option is not allowed for nested logit estimation. Instead, the CHOICE= option needs to be specified.

```
   proc mdc data=newdata;
      model decision = ttime / type=nlogit choice=(mode 1 2 3)
                               covest=hess;
      id pid;
      utility u(1,) = ttime;
      nest level(1) = (1 2 @ 1, 3 @ 2),
           level(2) = (1 2 @ 1);
   run;
```

In Figure 19.8, estimates of the inclusive value parameters, INC_L2G1C1 and INC_L2G1C2, are indicative of a nested model structure. See the section "Nested Logit" on page 965 and the section "Decision Tree and Nested Logit" on page 967 for more details on inclusive values.

```
                              The MDC Procedure

                            Nested Logit Estimates

                             Parameter Estimates

                                          Standard                Approx
  Parameter      DF      Estimate          Error     t Value    Pr > |t|

  ttime_L1        1       -0.4040         0.1241       -3.25      0.0011
  INC_L2G1C1      1        0.8016         0.4352        1.84      0.0655
  INC_L2G1C2      1        0.8087         0.3591        2.25      0.0243
```

Figure 19.8. Two-Level Nested Logit Estimates

The nested logit model is estimated with the restriction INC_L2G1C1=INC_L2G1C2 by specifying the SAMESCALE option. The estimation result is displayed in Figure 19.9.

```
   proc mdc data=newdata;
      model decision = ttime /
```

```
            type=nlogit choice=(mode 1 2 3)
            samescale covest=hess;
      id pid;
      utility u(1,) = ttime;
      nest level(1) = (1 2 @ 1, 3 @ 2),
           level(2) = (1 2 @ 1);
   run;
```

```
                              The MDC Procedure

                            Nested Logit Estimates

                             Parameter Estimates

                                            Standard                 Approx
   Parameter      DF      Estimate           Error    t Value     Pr > |t|

   ttime_L1        1       -0.4025          0.1217      -3.31       0.0009
   INC_L2G1        1        0.8209          0.3019       2.72       0.0066
```

Figure 19.9. Nested Logit Estimates with One Dissimilarity Parameter

The nested logit model is equivalent to the conditional logit model if INC_L2G1C1 = INC_L2G1C2 = 1. You can verify this relationship by estimating a constrained nested logit model. (See the "RESTRICT Statement" section on page 956 for details on imposing linear restrictions on parameter estimates.) The parameter estimates and the active linear constraints for the following constrained nested logit model are displayed in Figure 19.10.

```
   proc mdc data=newdata;
      model decision = ttime / type=nlogit
         choice=(mode 1 2 3) covest=hess;
      id pid;
      utility u(1,) = ttime;
      nest level(1) = (1 2 @ 1, 3 @ 2),
           level(2) = (1 2 @ 1);
      restrict INC_L2G1C1 = 1, INC_L2G1C2 =1;
   run;
```

```
                                The MDC Procedure

                              Nested Logit Estimates

                               Parameter Estimates

                                             Standard                  Approx
        Parameter      DF     Estimate          Error    t Value     Pr > |t|

        ttime_L1        1      -0.3572         0.0776      -4.60       <.0001
        INC_L2G1C1      0       1.0000              0
        INC_L2G1C2      0       1.0000              0
        Restrict1       1      -2.1706         8.4098      -0.26       0.7993*
        Restrict2       1       3.6573        10.0001       0.37       0.7186*

                               Parameter Estimates

                  Parameter     Parameter Label

                  ttime_L1
                  INC_L2G1C1
                  INC_L2G1C2
                  Restrict1      Linear EC [ 1 ]
                  Restrict2      Linear EC [ 2 ]

                 * Probability computed using beta distribution.


                  Linearly Independent Active Linear Constraints

                1          0  =   -1.0000  +    1.0000 * INC_L2G1C1
                2          0  =   -1.0000  +    1.0000 * INC_L2G1C2
```

Figure 19.10. Constrained Nested Logit Estimates

Multivariate Normal Utility Function

Consider the following random utility function:

$$U_{ij} = \text{ttime}_{ij}\beta + \epsilon_{ij}, \quad j = 1, 2, 3$$

where

$$\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \epsilon_{i3} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & \rho_{21} & 0 \\ \rho_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

The correlation coefficient (ρ_{21}) between U_{i1} and U_{i2} represents common neglected attributes of public transportation modes, 1 and 2. The following SAS statements estimate this trinomial probit model:

```
proc mdc data=newdata;
   model decision = ttime / type=mprobit nchoice=3
      unitvariance=(1 2 3) covest=hess;
   id pid;
run;
```

The UNITVARIANCE=(1 2 3) option specifies that the random component of utility for each of these choices has unit variance. If the UNITVARIANCE= option is specified, it needs to include at least two choices. The results of this constrained multinomial probit model estimation are displayed in Figure 19.11 and Figure 19.12. The test for ttime = 0 is rejected at the 1% significance level.

```
                         The MDC Procedure

                    Multinomial Probit Estimates

                         Model Fit Summary

              Dependent Variable                        decision
              Number of Observations                          50
              Number of Cases                                150
              Log Likelihood                           -33.88604
              Maximum Absolute Gradient                0.0002380
              Number of Iterations                             8
              Optimization Method               Dual Quasi-Newton
              AIC                                       71.77209
              Schwarz Criterion                         75.59613
```

Figure 19.11. Constrained Probit Estimation Summary

```
                         The MDC Procedure

                    Multinomial Probit Estimates

                        Parameter Estimates

                                          Standard                 Approx
    Parameter      DF       Estimate         Error     t Value    Pr > |t|

    ttime           1        -0.2307        0.0472       -4.89      <.0001
    RHO_21          1         0.4820        0.3135        1.54      0.1242
```

Figure 19.12. Multinomial Probit Estimates with Unit Variances

HEV and Multinomial Probit: Heteroscedastic Utility Function

When the stochastic components of utility are heteroscedastic and independent, you can model the data using an HEV or a multinomial probit model. The HEV model assumes that the utility of alternative j for each individual i has heteroscedastic random components.

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where the cumulative distribution function of the Gumbel distributed ϵ_{ij} is

$$F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}/\theta_j))$$

Note that the variance of ϵ_{ij} is $\frac{1}{6}\pi^2\theta_j^2$. Therefore, the error variance is proportional to the square of the scale parameter θ_j. For model identification, at least one of the scale parameters must be normalized to 1. The following SAS statements estimate an HEV model under a unit scale restriction for mode "1" ($\theta_1 = 1$):

```
proc mdc data=newdata;
   model decision = ttime / type=hev nchoice=3
      hev=(unitscale=1, integrate=laguerre)
      covest=hess;
   id pid;
run;
```

```
                          The MDC Procedure

              Heteroscedastic Extreme Value Model Estimates

                          Model Fit Summary

              Dependent Variable                       decision
              Number of Observations                         50
              Number of Cases                               150
              Log Likelihood                          -33.41383
              Maximum Absolute Gradient               0.0000218
              Number of Iterations                           11
              Optimization Method             Dual Quasi-Newton
              AIC                                      72.82765
              Schwarz Criterion                        78.56372
```

Figure 19.13. HEV Estimation Summary

```
                          The MDC Procedure

              Heteroscedastic Extreme Value Model Estimates

                         Parameter Estimates

                                          Standard                 Approx
    Parameter     DF      Estimate           Error    t Value    Pr > |t|

    ttime          1       -0.4407          0.1798      -2.45      0.0143
    SCALE2         1        0.7765          0.4348       1.79      0.0741
    SCALE3         1        0.5753          0.2752       2.09      0.0366
```

Figure 19.14. HEV Parameter Estimates

Note that the estimate of the HEV model is not always stable since computation of the log-likelihood function requires numerical integration. Bhat (1995) proposed the Gauss-Laguerre method. In general, the log-likelihood function value of HEV should be larger than that of conditional logit since HEV models include the conditional logit as a special case, but in this example the reverse is true (-33.414 for the HEV model, which is less than -33.321 for the conditional logit model). See Figure 19.13 and Figure 19.2. This indicates that the Gauss-Laguerre approximation to the true probability is too coarse. You can see how well the Gauss-Laguerre method works by specifying a unit scale restriction for all modes, since the HEV model with the unit variance for all modes reduces to the conditional logit model.

```
proc mdc data=newdata;
   model decision = ttime / type=hev nchoice=3
         hev=(unitscale=1 2 3, integrate=laguerre) covest=hess;
   id pid;
run;
```

Figure 19.15 shows that the ttime coefficient is not close to that of the conditional logit model.

```
                          The MDC Procedure

              Heteroscedastic Extreme Value Model Estimates

                         Parameter Estimates

                                         Standard                 Approx
    Parameter     DF     Estimate           Error    t Value    Pr > |t|

    ttime          1      -0.2926          0.0438      -6.68      <.0001
```

Figure 19.15. HEV Estimates with All Unit Scale Parameters

There is another option of specifying the integration method. The INTEGRATE=HARDY option uses the adaptive Romberg-type integration method. The adaptive integration produces much more accurate probability and log-likelihood function values, but often, it is not practical to use this method for analysis of the HEV model since it requires excessive CPU time. The following SAS statements produce the HEV estimates using the adaptive Romberg-type integration method. The results are displayed in Figure 19.16 and Figure 19.17.

```
proc mdc data=newdata;
   model decision = ttime / type=hev nchoice=3
      hev=(unitscale=1, integrate=hardy) covest=hess;
   id pid;
run;
```

```
                          The MDC Procedure

              Heteroscedastic Extreme Value Model Estimates

                          Model Fit Summary

              Dependent Variable                      decision
              Number of Observations                        50
              Number of Cases                              150
              Log Likelihood                         -33.02598
              Maximum Absolute Gradient              0.0001202
              Number of Iterations                           8
              Optimization Method            Dual Quasi-Newton
              AIC                                     72.05197
              Schwarz Criterion                       77.78803
```

Figure 19.16. HEV Estimation Summary Using Alternative Integration Method

```
                              The MDC Procedure

                 Heteroscedastic Extreme Value Model Estimates

                             Parameter Estimates

                                            Standard                 Approx
        Parameter     DF      Estimate         Error    t Value    Pr > |t|

        ttime          1       -0.4580        0.1861      -2.46      0.0139
        SCALE2         1        0.7757        0.4283       1.81      0.0701
        SCALE3         1        0.6908        0.3384       2.04      0.0412
```

Figure 19.17. HEV Estimates Using Alternative Integration Method

With the INTEGRATE=HARDY option, the log-likelihood function value of the HEV model, -33.026, is greater than that of the conditional logit model, -33.321. See Figure 19.16 and Figure 19.2.

When you impose unit scale restrictions on all choices, the HEV model gives the same estimates as the conditional logit model. See Figure 19.18 and Figure 19.5.

```
proc mdc data=newdata;
   model decision = ttime / type=hev nchoice=3
      hev=(unitscale=1 2 3, integrate=hardy) covest=hess;
   id pid;
run;
```

```
                              The MDC Procedure

                 Heteroscedastic Extreme Value Model Estimates

                             Parameter Estimates

                                            Standard                 Approx
        Parameter     DF      Estimate         Error    t Value    Pr > |t|

        ttime          1       -0.3572        0.0776      -4.60      <.0001
```

Figure 19.18. Alternative HEV Estimates with Unit Scale Restrictions

For comparison, we estimate a heteroscedastic multinomial probit model by imposing a zero restriction on the correlation parameter, $\rho_{31} = 0$. The MDC procedure requires normalization of at least two of the error variances in the multinomial probit model. Also, for identification, the correlation parameters associated with a unit normalized variance are restricted to be zero. When the UNITVARIANCE= option is specified, the zero restriction on correlation coefficients applies to the last choice of the list. In the following example, the variances of the first and second choices are normalized. The UNITVARIANCE=(1 2) option imposes additional restrictions that $\rho_{32} = \rho_{21} = 0$. The default for the UNITVARIANCE= option is the last two choices (which would have been equivalent to UNITVARIANCE=(2 3) for this example).

The utility function can be defined as

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where

$$\boldsymbol{\epsilon}_i \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}\right)$$

```
proc mdc data=newdata;
   model decision = ttime / type=mprobit nchoice=3
                            unitvariance=(1 2) covest=hess;
   id pid;
   restrict RHO_31 = 0;
run;
```

```
                            The MDC Procedure

                       Multinomial Probit Estimates

                           Parameter Estimates

                                         Standard                  Approx
Parameter      DF      Estimate           Error    t Value    Pr > |t|

ttime           1       -0.3206          0.0920      -3.49      0.0005
STD_3           1        1.6913          0.6906       2.45      0.0143
RHO_31          0             0               0
Restrict1       1        1.1854          1.5490       0.77      0.4499*

                           Parameter Estimates

          Parameter     Parameter Label

          ttime
          STD_3
          RHO_31
          Restrict1      Linear EC [ 1 ]

         * Probability computed using beta distribution.


          Linearly Independent Active Linear Constraints

          1         0  =         0  +    1.0000 * RHO_31
```

Figure 19.19. Heteroscedastic Multinomial Probit Estimates

Parameter Heterogeneity: Mixed Logit

One way of modeling unobserved heterogeneity across individuals in their sensitivity to observed exogenous variables is to use the mixed logit model with a random parameters or random coefficients specification. The probability of choosing alternative j is written

$$P_i(j) = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})}{\sum_{k=1}^{J} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}$$

where $\boldsymbol{\beta}$ is a vector of coefficients that varies across individuals, and $\mathbf{x}_{ij}$ is a vector of exogenous attributes.

For example, you can specify the distribution of the parameter β to be the normal distribution.

The mixed logit model uses a Monte Carlo simulation method to estimate the probabilities of choice. There are two simulation methods available. When the RANDNUM=PSEUDO option is given in the MODEL statement, pseudo-random numbers are generated, while the RANDNUM=HALTON option uses Halton quasi-random sequences. The default value is RANDNUM=HALTON.

You can estimate the model with normally distributed random coefficients of ttime with the following SAS statements:

```
proc mdc data=newdata type=mixedlogit;
   model decision = ttime / nchoice=3
         mixed=(normalparm=ttime);
   id pid;
run;
```

Let β^m and β^s be mean and scale parameters for the random coefficient, β. The relevant utility function is

$$U_{ij} = \text{ttime}_{ij}\beta + \epsilon_{ij}$$

where $\beta = \beta^m + \beta^s\eta$. β^m and β^s are fixed mean and scale parameters. The stochastic component, η, is assumed to be standard normal since the NORMALPARM= option is given. Alternatively, the UNIFORMPARM= or LOGNORMALPARM= option can be specified. The LOGNORMALPARM= option is useful when nonnegative parameters are being estimated. The NORMALPARM=, UNIFORMPARM=, and LOGNORMALPARM= variables must be included on the right-hand-side of the MODEL statement. See the "Mixed Logit Model" section on page 961 for more details. To estimate a mixed logit model using the transportation mode choice data, the MDC procedure requires the MIXED= option for random components. Results of the mixed logit estimation are displayed in Figure 19.20.

```
                         The MDC Procedure

                Mixed Multinomial Logit Estimates

                       Parameter Estimates

                                      Standard                 Approx
Parameter      DF      Estimate          Error    t Value    Pr > |t|

ttime_M         1       -0.5342         0.1861      -2.87      0.0041
ttime_S         1        0.2843         0.1715       1.66      0.0974
```

Figure 19.20. Mixed Logit Model Parameter Estimates

Syntax

The MDC procedure is controlled by the following statements:

PROC MDC *options* **;**
  BOUNDS *bound1 [, bound2 . . .]* **;**
  BY *variables* **;**
  ID *variable* **;**
  MODEL *dependent variables = regressors / options* **;**
  NEST *LEVEL(value) = ((values)@(value),. . ., (values)@(value))* **;**
  NLOPTIONS *options* **;**
  OUTPUT *options* **;**
  RESTRICT *restriction1 [, restriction2 . . .]* **;**
  UTILITY *U() = variables, . . ., U() = variables* **;**

Functional Summary

The statements and options used with the MDC procedure are summarized in the following table:

Description	Statement	Option
Data Set Options		
specify the input data set	MDC	DATA=
write parameter estimates to an output data set	MDC	OUTEST=
include covariances in the OUTEST= data set	MDC	COVOUT
write linear predictors and predicted probabilities to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
specify the ID variable	ID	
specify BY-group processing variables	BY	

Description	**Statement**	**Option**
Printing Control Options		
request all printing options	MODEL	ALL
display correlation matrix of the estimates	MODEL	CORRB
display covariance matrix of the estimates	MODEL	COVB
Model Estimation Options		
specify the choice variables	MODEL	CHOICE=()
specify the convergence criterion	MODEL	CONVERGE=
specify the type of covariance matrix	MODEL	COVEST=
specify the starting point of the Halton sequence	MODEL	HALTONSTART=
specify options specific to the HEV model	MODEL	HEV=()
set the initial values of parameters used by the iterative optimization algorithm	MODEL	INITIAL=()
specify the maximum number of iterations	MODEL	MAXITER=
specify the options specific to mixed logit	MODEL	MIXED=()
specify the number of choices for each person	MODEL	NCHOICE=
specify the number of simulations	MODEL	NSIMUL=
specify the optimization technique	MODEL	OPTMETHOD=
specify the type of random number generators	MODEL	RANDNUM=
specify that initial values are generated using random numbers	MODEL	RANDINIT
specify the rank dependent variable	MODEL	RANK
specify optimization restart options	MODEL	RESTART=()
specify a restriction on inclusive parameters	MODEL	SAMESCALE
specify a seed for pseudo-random number generation	MODEL	SEED=
specify a stated preference data restriction on inclusive parameters	MODEL	SPSCALE
specify the type of the model	MODEL	TYPE=
specify normalization restrictions on multinomial probit error variances	MODEL	UNITVARIANCE=()
NESTED Logit Related Options		
specify the tree structure	NEST	LEVEL()=
specify the type of utility function	UTILITY	U()=
Output Control Options		
output predicted values	OUTPUT	P=
output estimated linear predictor	OUTPUT	XBETA=

PROC MDC Statement

PROC MDC *options* **;**

The following options can be used in the PROC MDC statement.

DATA= *SAS-data-set*

specifies the input SAS data set. If the DATA= option is not specified, PROC MDC uses the most recently created SAS data set.

OUTEST= *SAS-data-set*

names the SAS data set that the parameter estimates are written to. See “OUTEST= Data Set” later in this chapter for information on the contents of this data set.

COVOUT

writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

In addition, any of the following MODEL statement options can be specified in the PROC MDC statement, which is equivalent to specifying the option for the MODEL statement: ALL, CONVERGE=, CORRB, COVB, COVEST=, HALTONSTART=, ITPRINT, MAXITER=, NOPRINT, NSIMUL=, OPTMETHOD=, RANDINIT, RANK, RESTART=, SAMESCALE, SEED=, SPSCALE, TYPE=, and UNITVARIANCE=.

BOUNDS Statement

BOUNDS *bound1 [, bound2 ...]* **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the MDC procedure. You can specify any number of BOUNDS statements.

Each *bound* is composed of variables, constants, and inequality operators:

item operator item [operator item [operator item . . .]]

Each *item* is a constant, the name of a regressor variable, or a list of regressor names. Each *operator* is ’<’, ’>’, ’<=’, or ’>=’.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See the “RESTRICT Statement” section on page 956 as well.

Lagrange multipliers are reported for all the active boundary constraints. In the displayed output, the Lagrange multiplier estimates are identified with the names Restrict1, Restrict2, and so forth. The probability of the Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive, or nonbinding, bounds have no effect on the estimation results and are not noted in the output.

The following BOUNDS statement constrains the estimates of the coefficient of ttime to be negative and the coefficients of x1 through x10 to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds ttime < 0,
       0 < x1-x10 < 1;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC MDC to obtain separate analyses on observations in groups defined by the BY variables.

ID Statement

ID *variable* **;**

The ID statement must be used with PROC MDC to specify the identification variable that controls multiple choice-specific cases. The MDC procedure requires only one ID statement even with multiple MODEL statements.

MODEL Statement

MODEL *dependent = regressors / options ;*

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. When the nested logit model is estimated, regressors in the UTILITY statement are used for estimation.

The following options can be used in the MODEL statement after a slash (/).

CHOICE= (*variables***)**
CHOICE=(*variable numbers***)**
specifies the variables that contain possible choices for each individual. Choice variables must have integer values. Multiple choice variables are only allowed for nested logit models. If all possible alternatives are written with the variable name, the MDC procedure checks all values of the choice variable. The CHOICE=(X 1 2 3) specification implies that the value of X should be 1, 2, or 3. On the other hand, the CHOICE=(X) specification considers all distinctive nonmissing values of X as elements of the choice set.

CONVERGE= *number*
specifies the convergence criterion. The CONVERGE= option is the same as the ABSGCONV= option in the NLOPTIONS statement. The ABSGCONV= option in the NLOPTIONS statement overrides the CONVERGE= option. The default value is 1E-5.

HALTONSTART= *number*

specifies the starting point of the Halton sequence. The specified number must be a positive integer. The default is HALTONSTART=11.

HEV= (*option-list***)**

specifies options that are used to estimate the HEV model. The HEV model with a unit scale for the alternative 1 is estimated using the following SAS statements:

```
model y = x1 x2 x3 / hev=(unitscale=1);
```

The following options can be used in the HEV=() option. These options are listed within parentheses and separated by commas.

INTORDER= *number*

specifies the number of summation terms for Gaussian quadrature integration. The default is INTORDER=40. The maximum order is limited to 45. This option applies only to the INTEGRATION=LAGUERRE method.

UNITSCALE= *number-list*

specifies restrictions on scale parameters of stochastic utility components.

INTEGRATE= LAGUERRE | HARDY

specifies the integration method. The INTEGRATE=HARDY option specifies an adaptive integration method, while the INTEGRATE=LAGUERRE option specifies the Gauss-Laguerre approximation method. The default is INTEGRATE=LAGUERRE.

MIXED= (*option-list***)**

specifies options that are used for mixed logit estimation. The mixed logit model with normally distributed random parameters is specified as follows:

```
model y = x1 x2 x3 / mixed=(normalparm=x1);
```

The following options can be used in the MIXED=() option. The options are listed within parentheses and separated by commas.

LOGNORMALPARM= *variables*

specifies the variables whose random coefficients are log-normally distributed. LOGNORMALPARM= variables must be included on the right-hand side of the MODEL statement.

NORMALEC= *variables*

specifies the error component variables whose coefficients have a normal distribution $N(0, \sigma^2)$.

NORMALPARM= *variables*

specifies the variables whose random coefficients are normally distributed. NORMALPARM= variables must be included on the right-hand side of the MODEL statement.

UNIFORMEC= *variables*

specifies the error component variables whose coefficients have a uniform distribution $U(-\sqrt{3}\sigma, \sqrt{3}\sigma)$.

UNIFORMPARM= *variables*

specifies the variables whose random coefficients are uniformly distributed. UNIFORMPARM= variables must be included on the right-hand side of the MODEL statement.

NCHOICE= *number*

specifies the number of choices for multinomial choice models when all individuals have the same choice set. The NCHOICE= and CHOICE= options must not be used simultaneously, and the NCHOICE= option cannot be used for nested logit models.

NSIMUL= *number*

specifies the number of simulations when the mixed logit or multinomial probit model is estimated. The default is NSIMUL=100. In general, you need a smaller number of simulations with RANDNUM=HALTON than with RANDNUM=PSEUDO.

RANDNUM= *value*

specifies the type of the random number generator used for simulation. RANDNUM=HALTON is the default. The following option values are allowed:

PSEUDO	specifies pseudo-random number generation
HALTON	specifies Halton sequence generation

RANDINIT
RANDINIT= *number*

specifies that initial parameter values are perturbed by uniform pseudo-random numbers for numerical optimization of the objective function. The default is $U(-1, 1)$. When the RANDINIT=r option is specified, $U(-r, r)$ pseudo-random numbers are generated. The value r should be positive. With a RANDINIT or RANDINIT= option, there are pure random searches for a given number of trials (1000 for conditional or nested logit, and 500 for other models) to get a maximum (or minimum) value of the objective function. For example, when there is a parameter estimate with an initial value of 1, the RANDINIT option will add a generated random number u to the initial value and compute an objective function value using $1 + u$. This option is helpful in finding the initial value automatically if there is no guidance in setting the initial estimate.

RANK

specifies that the dependent variable contains ranks. The numbers must be positive integers starting from 1. When the dependent variable has value 1, the corresponding alternative is chosen. This option is provided only as a convenience to the user: the extra information contained in the ranks is not currently used for estimation purposes.

RESTART=(*option-list*)

specifies options that are used for reiteration of the optimization problem. When the ADDRANDOM option is specified, the initial value of reiteration is computed using random grid searches around the initial solution.

```
   model y = x1 x2 / type=clogit
      restart=(addvalue=(.01 .01));
```

The preceding SAS statement reestimates a conditional logit model by adding ADDVALUE= values. If the ADDVALUE= option contains missing values, the restart option uses the corresponding estimate from the initial stage. If no ADDVALUE= value is specified for an estimate, a default value equal to (|estimate| * 1e-3) is added to the corresponding estimate from the initial stage. If both the ADDVALUE= and ADDRANDOM(=) options are specified, ADDVALUE= is ignored.

The following options can be used in the RESTART=() option. The options are listed within parentheses.

ADDMAXIT= *number*

specifies the maximum number of iterations for the second stage of the estimation. The default is ADDMAXIT=100.

ADDRANDOM

ADDRANDOM= *value*

specifies random added values to the estimates from the initial stage. With the ADDRANDOM option, $U(-1, 1)$ random numbers are created and added to the estimates obtained in the initial stage. When the ADDRANDOM=r option is specified, $U(-r, r)$ random numbers are generated. The restart initial value is determined based on the given number of random searches (1000 for conditional or nested logit, and 500 for other models).

ADDVALUE= (*value-list*)

specifies values added to the estimates from the initial stage. A missing value in the list is considered as a zero value for the corresponding estimate. When the ADDVALUE= option is not specified, default values equal to (|estimate| * 1e-3) are added.

SAMESCALE

specifies that the parameters of the inclusive values are the same within a group at each level when nested logit is estimated.

SEED= *number*

specifies an initial seed for pseudo-random number generation. The SEED= value must be less than $2^{31} - 1$. If the SEED= value is negative or zero, the time of day from the computer's clock is used to obtain the initial seed. The default is SEED=0.

SPSCALE

specifies that the parameters of the inclusive values are the same for any choice with only one nested choice within a group, for each level in a nested logit model. This option is useful in analyzing stated preference data.

TYPE= *value*

specifies the type of model to be analyzed. The supported model types are

CONDITIONLOGIT | CLOGIT | CL	specifies a conditional logit model
HEV	specifies a heteroscedastic extreme value model
MIXEDLOGIT | MXL	specifies a mixed logit model
MULTINOMPROBIT | MPROBIT | MP	specifies a multinomial probit model
NESTEDLOGIT | NLOGIT | NL	specifies a nested logit model

UNITVARIANCE= (*number-list***)**

specifies normalization restrictions on error variances of multinomial probit for the choices whose numbers are given in the list. If the UNITVARIANCE= option is specified, it must include at least two choices. Also, for identification, additional zero restrictions are placed on the correlation coefficients for the last choice in the list.

Printing Options

ALL

requests all printing options.

COVB

displays the estimated covariances of the parameter estimates.

CORRB

displays the estimated correlation matrix of the parameter estimates.

COVEST= *value*

The COVEST= option specifies the type of covariance matrix. Possible values are OP, HESSIAN, and QML. When COVEST=OP is specified, the outer product matrix is used to compute the covariance matrix of the parameter estimates. The COVEST=HESSIAN option produces the covariance matrix using the inverse Hessian matrix. The quasi-maximum likelihood estimates are computed with COVEST=QML. The default is COVEST=HESSIAN when the Newton-Raphson method is used. COVEST=OP is the default when the OPTMETHOD=QN option is specified. The supported covariance types are

OP	specifies the covariance from the outer product matrix
HESSIAN	specifies the covariance from the Hessian matrix
QML	specifies the covariance from the outer product and Hessian matrices

ITPRINT
displays the initial parameter estimates, convergence criteria, and constraints of the optimization. At each iteration, objective function value, maximum absolute gradient element, step size, and slope of search direction are printed as well. The objective function is the full negative log-likelihood function for the maximum likelihood method. When the ITPRINT option is specified in the presence of the NLOPTIONS statement, all printing options in the NLOPTIONS statement are ignored.

NOPRINT
suppresses all displayed output.

Estimation Control Options

You can also specify detailed optimization options in the NLOPTIONS statement. The OPTMETHOD= option overrides the TECHNIQUE= option in the NLOPTIONS statement. Note that the NLOPTIONS statement is ignored if the OPTMETHOD= option is specified.

INITIAL= *(initial-values)*
START= *(initial-values)*
specifies initial values for some or all of the parameter estimates. The values specified are assigned to model parameters in the same order as the parameter estimates are displayed in the MDC procedure output.

When you use the INITIAL= option, the initial values in the INITIAL= option must satisfy the restrictions specified for the parameter estimates. If they do not, the initial values you specify are adjusted to satisfy the restrictions.

MAXITER= *number*
sets the maximum number of iterations allowed. The MAXITER= option overrides the MAXITER= option in the NLOPTIONS statement. The default is MAXITER=100.

OPTMETHOD= *value*
The OPTMETHOD= option specifies the optimization technique when the estimation method uses nonlinear optimization.

QN	specifies the quasi-Newton method
NR	specifies the Newton-Raphson method
TR	specifies the trust region method

The OPTMETHOD=NR option is the same as the TECHNIQUE=NEWRAP option in the NLOPTIONS statement. For the conditional and nested logit models the default is OPTMETHOD=NR. For other models OPTMETHOD=QN is the default.

NEST Statement

NEST *LEVEL(level number)= (choices@choice, . . .)* **;**

The NEST statement is used when one choice variable contains all possible alternatives and the TYPE=NLOGIT option is specified. The decision tree is constructed based on the NEST statement. When the choice set is specified using multiple CHOICE= variables in the MODEL statement, the NEST statement is ignored.

Consider the following eight choices that are nested in a three-level tree structure.

```
Level 1     Level 2     Level 3    top
   1           1           1        1
   2           1           1        1
   3           1           1        1
   4           2           1        1
   5           2           1        1
   6           2           1        1
   7           3           2        1
   8           3           2        1
```

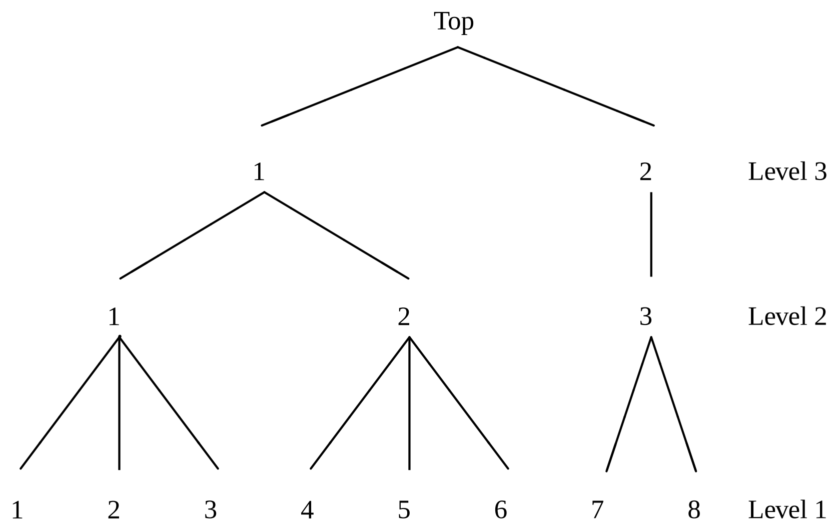


Figure 19.21. Three-Level Tree

You can use the NEST statement to specify the tree structure displayed in Figure 19.21.

```
nest level(1) = (1 2 3 @ 1, 4 5 6 @ 2, 7 8 @ 3),
     level(2) = (1 2 @ 1, 3 @ 2),
     level(3) = (1 2 @ 1);
```

Note that the decision tree is constructed based on the sequence of first-level choice set specification. Therefore, specifying another order at Level 1 builds a different tree. The following NEST statement builds the tree displayed in Figure 19.22.

```
nest level(1) = (4 5 6 @ 2, 1 2 3 @ 1, 7 8 @ 3),
     level(2) = (1 2 @ 1, 3 @ 2),
     level(3) = (1 2 @ 1);
```

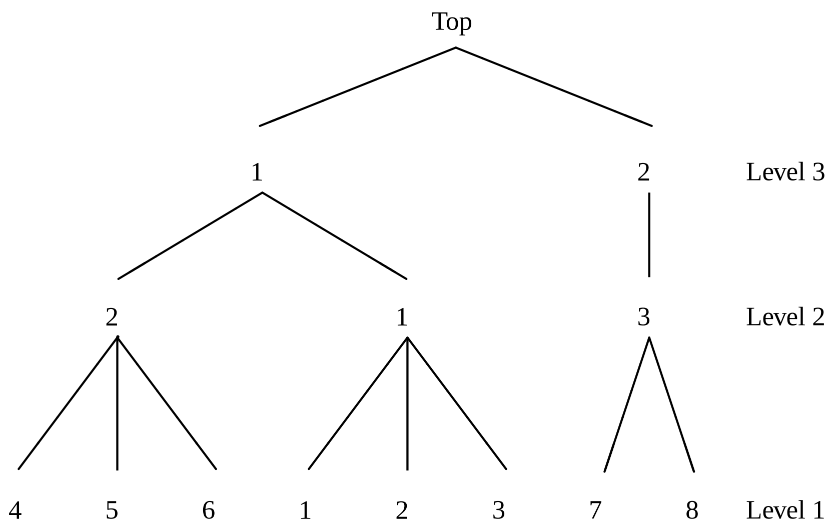


Figure 19.22. Alternative Three-Level Tree

However, the SAS statement with a different sequence of choice specification at higher levels builds the same tree as displayed in Figure 19.21 if the sequence at the first level is the same.

```
   nest level(1) = (1 2 3 @ 1, 4 5 6 @ 2, 7 8 @ 3),
        level(2) = (3 @ 2, 1 2 @ 1),
        level(3) = (1 2 @ 1);
```

The following specifications are equivalent:

```
   nest level(2) = (3 @ 2, 1 2 @ 1)

   nest level(2) = (3 @ 2, 1 @ 1, 2 @ 1)

   nest level(2) = (1 @ 1, 2 @ 1, 3 @ 2)
```

Since the MDC procedure contains multiple cases for each individual, it is important to keep data sequence in proper order. Consider the four-choice multinomial model with one explanatory variable cost.

```
   pid   choice  y   cost
    1      1     1    10
    1      2     0    25
    1      3     0    20
    1      4     0    30
    2      1     0    15
    2      2     0    22
    2      3     1    16
    2      4     0    25
```

The order of data needs to correspond to the value of choice. Therefore, the following data set is equivalent to the preceding data.

```
pid   choice  y   cost
 1      2     0    25
 1      3     0    20
 1      1     1    10
 1      4     0    30
 2      3     1    16
 2      4     0    25
 2      1     0    15
 2      2     0    22
```

The two-level nested model is estimated with a NEST statement.

```
proc mdc data=one type=nlogit;
     model y = cost / choice=(choice);
     id pid;
     utility(1,) = cost;
     nest level(1) = (1 2 3 @ 1, 4 @ 2),
          level(2) = (1 2 @ 1);
```

The tree is constructed as in Figure 19.23.

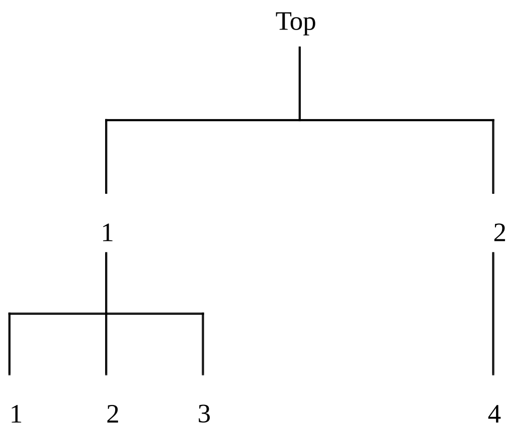


Figure 19.23. Two-Level Tree

Of course, another model is estimated if you specify the decision tree as in Figure 19.24. The different nested tree structure is specified in the following SAS statement:

```
proc mdc data=one type=nlogit;
     model y = cost / choice=(choice);
     id pid;
     utility u(1,) = cost;
     nest level(1) = (1 @ 1, 2 3 4 @ 2),
          level(2) = (1 2 @ 1);
```

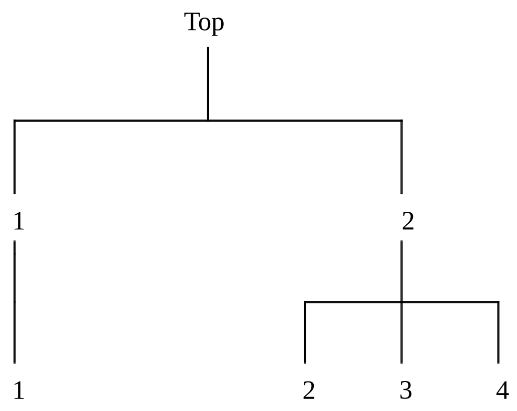


Figure 19.24. Two-Level Tree

NLOPTIONS Statement

NLOPTIONS *options* **;**

PROC MDC uses the NonLinear Optimization (NLO) subsystem to perform nonlinear optimization tasks. The NLOPTIONS statement specifies nonlinear optimization options. The NLOPTIONS statement must follow the MODEL statement. For a list of all the options of the NLOPTIONS statement, see Chapter 10, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT *options* **;**

The MDC procedure supports the OUTPUT statement. The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimated linear predictors (XBETA) and predicted probabilities (P). The input data set must be sorted by the choice variable(s) within each ID.

OUT= *SAS-data-set*
specifies the name of the output data set.

PRED= *variable name*
P= *variable name*
requests the predicted probabilities by naming the variable that contains the predicted probabilities in the output data set.

XBETA= *variable name*
names the variable that contains the linear predictor ($\mathbf{x}'\boldsymbol{\beta}$) values. However, the XBETA= option is not supported in the nested logit model.

RESTRICT Statement

RESTRICT *restriction1 [, restriction2 ...]* **;**

The RESTRICT statement is used to impose linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <= , or >=.

Restriction expressions can be composed of variable names, times ($*$) and plus ($+$) operators, and constants. Variables named in restriction expressions must be among the variables estimated by the model. The restriction expressions must be a linear function of the variables.

Lagrange multipliers are reported for all the active linear constraints. In the displayed output, the Lagrange multiplier estimates are identified with the names Restrict1, Restrict2, and so forth. The probability of the Lagrange multipliers are computed using a beta distribution (LaMotte 1994).

The following is an example of the use of the RESTRICT statement:

```
proc mdc data=one;
model y = x1-x10 /  type=clogit
   choice=(mode 1 2 3);
id pid;
restrict x1*2 <= x2 + x3;
run;
```

UTILITY Statement

UTILITY *U(level, <choices>)= variables ;*

The UTILITY statement can be used in estimating a nested logit model. The U()= option can have two arguments. The first argument contains level information while the second argument is related to choice information. The second argument can be omitted for the first level when all the choices at the first level share the same variables. The UTILITY statement specifies a utility function while the NEST statement constructs the decision tree.

Consider a two-level nested logit model that has one explanatory variable at level 1. This model can be specified as

```
proc mdc data=one type=nlogit;
     model y = cost / choice=(choice);
     id pid;
     utility u(1,2 3 4) = cost;
     nest level(1) = (1 @ 1, 2 3 4 @ 2),
          level(2) = (1 2 @ 1);
```

Of course, you also can specify

```
   utility u(1,) = cost;
```

since all the variables at the first level share the same explanatory variable, cost. The variable, cost, should be listed in the MODEL statement. When the additional explanatory variable, dummy, is included at level 2, another U()= option needs to be specified. Note that the U()= option must specify choices within any level above the first. Thus, it is specified as U(2, 1 2) below.

```
   proc mdc data=one type=nlogit;
        model y = cost dummy/ choice=(choice);
        id pid;
        utility u(1,) = cost,
                u(2,1 2) = dummy;
        nest level(1) = (1 @ 1, 2 3 4 @ 2),
             level(2) = (1 2 @ 1);
```

Details

Multinomial Discrete Choice Modeling

When the dependent variable takes multiple discrete values, you can use multinomial discrete choice modeling to analyze the data. We consider models for unordered multinomial data. Let the random utility function be defined by

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where the subscript i is an index for the individual, the subscript j is an index for the alternative, V_{ij} is a nonstochastic utility function, and ϵ_{ij} is a random component, or error, which captures unobserved characteristics of alternatives and/or individuals. In multinomial discrete choice models, the utility function is assumed to be linear, so that $V_{ij} = \mathbf{x}_{ij}'\boldsymbol{\beta}$. In the conditional logit model, each ϵ_{ij} for all $j \in C_i$ is distributed independently and identically (iid) with the type I extreme value distribution, $\exp(-\exp(-\epsilon_{ij}))$, also known as the Gumbel distribution. The iid assumption on the random components of the utilities of the different alternatives can be relaxed to overcome the well-known and restrictive *Independence from Irrelevant Alternatives* (IIA) property of the conditional logit model. This allows for more flexible substitution patterns among alternatives than the one imposed by the conditional logit model. See the section "Independence from Irrelevant Alternatives (IIA)" on page 960. The nested logit model is derived by allowing the random components to be identical but nonindependent. Instead of independent type I extreme value errors, the errors are assumed to have a generalized extreme value distribution. This model generalizes the conditional logit model to allow for particular patterns of correlation in unobserved utility (McFadden 1978). Another generalization of the conditional logit model, the heteroscedastic extreme value (HEV) model, is obtained by allowing independent but nonidentical errors distributed with a type I extreme value distribution (Bhat 1995). It permits different variances on the random components of utility across the alternatives. Mixed logit models are also generalizations of the conditional logit model that can represent very general patterns of substitution among alternatives. See the “Mixed Logit Model” section on page 961 for details. The multinomial probit (MNP) model is derived when the errors, $(\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{iJ})$, have a multivariate normal (MVN) distribution. Thus, this model accommodates a very general error structure.

The multinomial probit model requires burdensome computation compared to a family of multinomial choice models derived from the Gumbel distributed utility function, since it involves multi-dimensional integration (with dimension $J - 1$) in the estimation process. In addition, the multinomial probit model requires more parameters than other multinomial choice models. As a result, conditional and nested logit models are used more frequently, even though they are derived from a utility function whose random component is more restrictively defined than the multinomial probit model.

The event of a choice being made, $\{y_i = j\}$, can be expressed using a random utility function as follows:

$$U_{ij} \geq \mathbf{max}_{k \in C_i, k \neq j} U_{ik}$$

where C_i is the choice set of individual i. Individual i chooses alternative j if and only if it provides a level of utility that is greater than or equal to that of any other alternative in his choice set. Then, the probability that individual i chooses alternative j (from among the n_i choices in his choice set C_i) is

$$P_i(j) = P_{ij} = P[\mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij} \geq \mathbf{max}_{k \in C_i}(\mathbf{x}'_{ik}\boldsymbol{\beta} + \epsilon_{ik})] \tag{19.1}$$

Multinomial Logit and Conditional Logit

When explanatory variables contain only individual characteristics, the multinomial logit model is defined as

$$P(y_i = j) = P_{ij} = \frac{\exp(\mathbf{x}'_i\boldsymbol{\beta}_j)}{\sum_{k=0}^{J} \exp(\mathbf{x}'_i\boldsymbol{\beta}_k)} \quad \text{for } j = 0, \cdots, J$$

where y_i is a random variable that indicates the choice made, x_i is a vector of characteristics specific to the ith individual, and $\boldsymbol{\beta}_j$ is a vector of coefficients specific to the jth alternative. Thus, this model involves choice specific coefficients and only individual specific regressors. For model identification, one often assumes that $\boldsymbol{\beta}_0 = 0$. The multinomial logit model reduces to the binary logit model if $J = 1$.

The ratio of the choice probabilities for alternatives j and l, or the *odds ratio* of alternatives j and l, is

$$\frac{P_{ij}}{P_{il}} = \frac{\exp(\mathbf{x}'_i\boldsymbol{\beta}_j)/\sum_{k=0}^{J}\exp(\mathbf{x}'_i\boldsymbol{\beta}_k)}{\exp(\mathbf{x}'_i\boldsymbol{\beta}_l)/\sum_{k=0}^{J}\exp(\mathbf{x}'_i\boldsymbol{\beta}_k)} = \exp[\mathbf{x}'_i(\boldsymbol{\beta}_j - \boldsymbol{\beta}_l)]$$

Note that the odds ratio of alternatives j and l does not depend on any alternatives other than j and l. For more on this, see the section "Independence from Irrelevant Alternatives (IIA)" on page 960.

The log-likelihood function of the multinomial logit model is

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=0}^{J} d_{ij} \ln P(y_i = j)$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

This type of multinomial choice modeling has a couple of weaknesses: it has too many parameters (the number of individual characteristics times J) and it is difficult to interpret. The multinomial logit model can be used to predict the choice probabilities, among a given set of $J+1$ alternatives, of an individual with known vector of characteristics $\mathbf{x}_i$.

The parameters of the multinomial logit model can be estimated with the TYPE=CLOGIT option in the MODEL statement; however, this requires modification of the conditional logit model to allow individual specific effects.

The conditional logit model, sometimes also called the multinomial logit model, is similarly defined when choice specific data are available. Using properties of type I extreme value (or Gumbel) distribution, the probability that individual i chooses alternative j from among the choices in his choice set C_i is

$$P(y_i = j) = P_{ij} = P[\mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij} \geq \max_{k \in C_i, k \neq j}(\mathbf{x}'_{ik}\boldsymbol{\beta} + \epsilon_{ik})] = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})}{\sum_{k \in C_i} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}$$

where $\mathbf{x}_{ij}$ is a vector of attributes specific to the jth alternative as perceived by the ith individual. It is assumed that there are n_i choices in each individual's choice set, C_i.

The log-likelihood function of the conditional logit model is

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j \in C_i} d_{ij} \ln P(y_i = j)$$

The conditional logit model can be used to predict the probability that an individual will choose a previously unavailable alternative, given knowledge of $\boldsymbol{\beta}$ and the vector $\mathbf{x}_{ij}$ of choice specific characteristics.

Independence from Irrelevant Alternatives (IIA)

The problematic aspect of the conditional logit (and the multinomial logit) model lies in the property of independence from irrelevant alternatives (IIA). The IIA property can be derived from the probability ratio of any two choices. For the conditional logit model,

$$\frac{P_{ij}}{P_{il}} = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})/\sum_{k\in C_i}\exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}{\exp(\mathbf{x}'_{il}\boldsymbol{\beta})/\sum_{k\in C_i}\exp(\mathbf{x}'_{ik}\boldsymbol{\beta})} = \exp[(\mathbf{x}_{ij}-\mathbf{x}_{il})'\boldsymbol{\beta}]$$

It is evident that the ratio of the probabilities for alternatives j and l does not depend on any alternatives other than j and l. This was also shown to be the case for the multinomial logit model. Thus, for the conditional and multinomial logit models, the ratio of probabilities of any two alternatives is necessarily the same regardless of what other alternatives are in the choice set or what the characteristics of the other alternatives are. This is referred to as the IIA property.

The IIA property is useful from the point of view of estimation and forecasting. For example, it allows the prediction of demand for currently unavailable alternatives. If the IIA property is appropriate for the choice situation being considered, then estimation can be based on the set of currently available alternatives and then the estimated model can be used to calculate the probability that an individual would choose a new alternative not considered in the estimation procedure. However, the IIA property is restrictive from the point of view of choice behavior. Models that display the IIA property predict that a change in the attributes of one alternative changes the probabilities of the other alternatives proportionately such that the ratios of probabilities remain constant. Thus, cross elasticities due to a change in the attributes of an alternative j are equal for all alternatives $k \neq j$. This particular substitution pattern might be too restrictive in some choice settings.

The IIA property of the conditional logit model follows from the assumption that the random components of utility are identically and independently distributed. The other models in PROC MDC, namely, nested logit, HEV, mixed logit, and multinomial probit, relax the IIA property in different ways.

Heteroscedastic Extreme Value Model

The heteroscedastic extreme value (HEV) model (Bhat 1995) allows the random components of the utility function to be nonidentical. Specifically, the HEV model assumes independent but nonidentical error terms distributed with the type I extreme value distribution. The HEV model allows the variances of the random components of utility to differ across alternatives. Bhat (1995) argues that the HEV model does not have the IIA property. The HEV model contains the conditional logit model as a special case. The probability that an individual i will choose alternative j from the set C_i of available alternatives is

$$P_i(j) = \int_{-\infty}^{\infty} \prod_{k\in C_i, k\neq j} \Gamma\left[\frac{\mathbf{x}'_{ij}\boldsymbol{\beta} - \mathbf{x}'_{ik}\boldsymbol{\beta} + \theta_j w}{\theta_k}\right]\gamma(w)dw$$

where the choice set C_i has n_i elements and

$$\Gamma(x) = \exp(-\exp(-x))$$

$$\gamma(x) = \exp(-x)\Gamma(x)$$

are the cumulative distribution function and probability density function of the type I extreme value distribution. The variance of the error term for the jth alternative is $\frac{1}{6}\pi^2\theta_j^2$. If the scale parameters, θ_j, of the random components of utility of all alternatives are equal, then this choice probability is the same as that of the conditional logit model. The log-likelihood function of the HEV model can be written as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j \in C_i} d_{ij} \ln[P_i(j)]$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

Since the log-likelihood function contains an improper integral function, it is computationally difficult to get a stable estimate. With the transformation $u = \exp(-w)$, the probability can be written

$$\begin{aligned} P_i(j) &= \int_0^\infty \Pi_{k \in C_i, k \neq j} \Gamma \left[\frac{\mathbf{x}'_{ij}\boldsymbol{\beta} - \mathbf{x}'_{ik}\boldsymbol{\beta} - \theta_j \ln(u)}{\theta_k} \right] \exp(-u)du \\ &= \int_0^\infty G_{ij}(u) \exp(-u)du \end{aligned}$$

Using the Gauss-Laguerre weight function, $W(x) = \exp(-u)$, the integration of the log-likelihood function can be replaced with a summation as follows:

$$\int_0^\infty G_{ij}(u) \exp(-u)du = \sum_{k=1}^{K} w_k G_{ij}(x_k)$$

Weights (w_k) and abscissas (x_k) are tabulated by Abramowitz and Stegun (1970).

Mixed Logit Model

In mixed logit models, an individual's utility from any alternative can be decomposed into a deterministic component, $\mathbf{x}'_{ij}\boldsymbol{\beta}$, which is a linear combination of observed variables, and a stochastic component, $\xi_{ij} + \epsilon_{ij}$.

$$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij} + \epsilon_{ij}$$

where $\mathbf{x}_{ij}$ is a vector of observed variables relating to individual i and alternative j, $\boldsymbol{\beta}$ is a vector of parameters, ξ_{ij} is an error component that can be correlated among alternatives and heteroscedastic for each individual, and ϵ_{ij} is a random term with zero mean that is independently and identically distributed over alternatives and individuals. The conditional logit model is derived if you assume ϵ_{ij} has an iid Gumbel distribution and $V(\xi_{ij}) = 0$.

The mixed logit model assumes a general distribution for ξ_{ij} and an iid Gumbel distribution for ϵ_{ij}. Denote the density function of the error component ξ_{ij} as $f(\xi_{ij}|\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is a parameter vector of the distribution of ξ_{ij}. The choice probability of alternative j for individual i is written as

$$P_i(j) = \int Q_i(j|\xi_{ij}) f(\xi_{ij}|\boldsymbol{\gamma}) d\xi_{ij}$$

where the conditional choice probability for a given value of ξ_{ij} is logit:

$$Q_i(j|\xi_{ij}) = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij})}{\sum_{k \in C_i} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta} + \xi_{ik})}$$

Since ξ_{ij} is not given, the unconditional choice probability, $P_i(j)$, is the integral of the conditional choice probability, $Q_i(j|\xi_{ij})$, over the distribution of ξ_{ij}. This model is called "mixed logit" since the choice probability is a mixture of logits with $f(\xi_{ij}|\boldsymbol{\gamma})$ as the mixing distribution.

In general, the mixed logit model does not have an exact likelihood function since the probability $P_i(j)$ does not always have a closed form solution. Therefore, a simulation method is used for computing the approximate probability.

$$\tilde{P}_i(j) = 1/S \sum_{s=1}^{S} \tilde{Q}_i(j|\xi_{ij}^s)$$

where S is the number of simulation replications and $\tilde{P}_i(j)$ is a simulated probability. The simulated log-likelihood function is computed as

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} d_{ij} \ln(\tilde{P}_i(j))$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

For simulation purposes, assume that the error component has a specific structure

$$\xi_{ij} = \mathbf{z}'_{ij}\boldsymbol{\mu} + \mathbf{w}'_{ij}\boldsymbol{\beta}^*$$

where $\mathbf{z}_{ij}$ is a vector of observed data and $\boldsymbol{\mu}$ is a random vector with zero mean and density function $\psi(\boldsymbol{\mu}|\boldsymbol{\gamma})$. The observed data vector ($\mathbf{z}_{ij}$) of the error component may contain some or all elements of $\mathbf{x}_{ij}$. The component $\mathbf{z}'_{ij}\boldsymbol{\mu}$ induces heteroscedasticity and correlation across unobserved utility components of the alternatives. This allows flexible substitution patterns among the alternatives. The kth element of vector $\boldsymbol{\mu}$ is distributed as

$$\mu_k \sim (0, \sigma_k^2)$$

Therefore, μ_k can be specified as

$$\mu_k = \sigma_k \epsilon_\mu$$

where

$$\epsilon_\mu \sim N(0,1)$$

or

$$\epsilon_\mu \sim U(-\sqrt{3}, \sqrt{3})$$

In addition, $\boldsymbol{\beta}^*$ is a vector of random parameters (or, random coefficients). Random coefficients allow heterogeneity across individuals in their sensitivity to observed exogenous variables. The observed data vector, $\mathbf{w}_{ij}$, is a subset of $\mathbf{x}_{ij}$. Three types of distributions for the random coefficients are supported. Denote the mth element of $\boldsymbol{\beta}^*$ as β^*_m.

- Normally distributed coefficient with the mean b_m and standard deviation s_m being estimated.

  $$\beta^*_m = b_m + s_m\epsilon_\beta \quad \text{and} \quad \epsilon_\beta \sim N(0,1)$$

- Uniformly distributed coefficient with the mean b_m and spread s_m being estimated. A uniform distribution with mean b and spread s is $U(b-s, b+s)$.

  $$\beta^*_m = b_m + s_m\epsilon_\beta \quad \text{and} \quad \epsilon_\beta \sim U(-1,1)$$

- Log-normally distributed coefficient with the mean b_m and standard deviation s_m being estimated. The coefficient is calculated as

  $$\beta^*_m = \exp(m + s\epsilon_\beta) \quad \text{and} \quad \epsilon_\beta \sim N(0,1)$$

  where m and s are parameters that are estimated. The mean and standard deviation of the log-normally distributed coefficient is then calculated as

  $$b_m = \exp[m + \frac{1}{2}s^2]$$

  and

  $$s_m = b_m[\exp(s^2) - 1]^{\frac{1}{2}}$$

A detailed description of mixed logit models can be found, for example, in Brownstone and Train (1999).

Multinomial Probit

The multinomial probit model allows the random components of the utility of the different alternatives to be nonindependent and nonidentical. Thus, it does not have the IIA property. The increase in the flexibility of the error structure comes at the expense of introducing several additional parameters in the covariance matrix of the errors.

Consider the random utility function

$$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$$

where the joint distribution of $(\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{iJ})$ is multivariate normal:

$$\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \vdots \\ \epsilon_{iJ} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = [\sigma_{jk}]_{j,k=1,\ldots,J}$$

The dimension of the error covariance matrix is determined by the number of alternatives J. Given $(\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{iJ})$, the jth alternative is chosen if and only if $U_{ij} \geq U_{ik}$ for all $k \neq j$. Thus, the probability that the jth alternative is chosen is

$$P(y_i = j) = P_{ij} = P[\epsilon_{i1} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{i1})'\boldsymbol{\beta}, \ldots, \epsilon_{iJ} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{iJ})'\boldsymbol{\beta}]$$

where y_i is a random variable that indicates the choice made. This is a cumulative probability from a $(J-1)$-variate normal distribution. Since evaluation of this probability involves multidimensional integration, it is practical to use a simulation method to estimate the model. Many studies have shown that the simulators proposed by Geweke (1989), Hajivassiliou (1993), and Keane (1994) (GHK) perform well. For example, Hajivassiliou et al. (1996) compare 13 simulators using 11 different simulation methods and conclude that the GHK simulation method is the most reliable. To compute the probability of the multivariate normal distribution, the recursive simulation method is used. Refer to Hajivassiliou (1993) for more details on GHK simulators.

The log-likelihood function for the multinomial probit model can be written as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{J} d_{ij} \ln P(y_i = j)$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

For identification of the multinomial probit model, two of the diagonal elements of Σ are normalized to 1 and it is assumed that for one of the choices whose error variance is normalized to 1, say k, it is also true that $\sigma_{jk} = \sigma_{kj} = 0$ for $j = 1, \cdots, J$ and $j \neq k$. Thus, a model with J alternatives has at most $J(J-1)/2 - 1$ covariance parameters after normalization.

Let D and R be defined as

$$D = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_J \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1J} \\ \rho_{21} & 1 & \cdots & \rho_{2J} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{J1} & \rho_{J2} & \cdots & 1 \end{bmatrix}$$

where $\sigma_j^2 = \sigma_{jj}$ and $\rho_{jk} = \frac{\sigma_{jk}}{\sigma_j \sigma_k}$. Then, for identification, $\sigma_{J-1} = \sigma_J = 1$ and $\rho_{kJ} = \rho_{Jk} = 0$, for all $k \neq J$ can be imposed, and the error covariance matrix is $\Sigma = DRD$.

In principle, the multinomial probit model is fully identified with the above normalizations, however, in practice, convergence in applications of the model with more than three alternatives often requires additional restrictions on the elements of $\mathbf{\Sigma}$.

It must also be noted that the unrestricted structure of the error covariance matrix makes it impossible to forecast demand for a new alternative without knowledge of the new $(J+1)$ by $(J+1)$ error covariance matrix.

Nested Logit

The nested logit model (McFadden 1978, 1981) allows partial relaxation of the assumption of independence of the stochastic components of utility of alternatives. In some choice situations, the IIA property holds for some pairs of alternatives but not all. In these situations, the nested logit model can be used if the set of alternatives faced by an individual can be partitioned into subsets such that the IIA property holds within subsets but not across subsets.

In the nested logit model, the joint distribution of the errors is generalized extreme value (GEV). This is a generalization of the type I extreme value distribution that gives rise to the conditional logit model. Note that all ϵ_{ij} within each subset are correlated with each other. Refer to McFadden (1978, 1981) for details.

Nested logit models can be described analytically following the notation of McFadden (1981). Assume that there are L levels, with 1 representing the lowest and L representing the highest level of the tree. The index of a node at level h in the tree is a pair (j_h, π_h), where $\pi_h = (j_{h+1}, \cdots, j_L)$ is the index of the adjacent node at level $h+1$. Thus, the primitive alternatives, at level 1 in the tree, are indexed by vectors $(j_1, \cdots, j_L)$, while the alternative nodes at level L are indexed by integers j_L. The choice set C_{π_h} is the set of primitive alternatives (at level 1) that belong to branches below the node π_h. The notation C_{π_h} is also used to denote a set of indices j_h such that (j_h, π_h) is a node immediately below π_h. Note that C_{π_0} is a set with a single element, while C_{π_L} represents a choice set containing all possible alternatives. As an example, consider the circled node at level 1 in Figure 19.25. Since it stems from node 11, $\pi_h = 11$, and, since it is the second node stemming from 11, $j_h = 2$, so that its index is $\pi_{h-1} = \pi_0 = (j_h, \pi_h) = 211$. Similarly, $C_{11} = \{111, 211, 311\}$ contains all the possible choices below 11.

Note that while this notation is useful for writing closed form solutions for probabilities, the MDC procedure allows a more flexible definition of indices. See the NEST statement in the "Syntax" section for more details on how to describe decision trees within the MDC procedure.

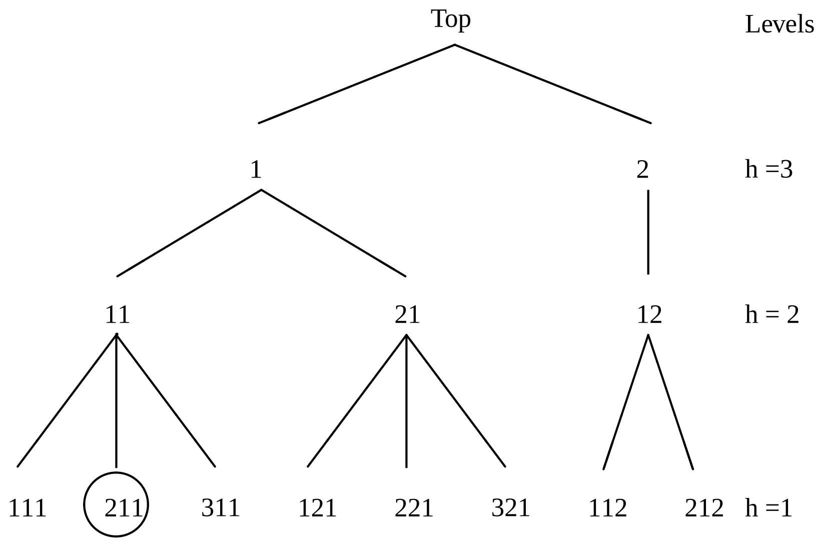


Figure 19.25. Node Indices for a Three-Level Tree

Let $\mathbf{x}^{(h)}_{i;j_h\pi_h}$ denote the vector of observed variables for individual i common to the alternatives below node $j_h\pi_h$. The probability of choice at level h has a closed form solution and is written

$$P_i(j_h|\pi_h) = \frac{\exp\left[\mathbf{x}^{(h)\prime}_{i;j_h\pi_h}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j_h\pi_h}} I_{k,j_h\pi_h}\theta_{k,j_h\pi_h}\right]}{\sum_{j\in C_{i;\pi_h}} \exp\left[\mathbf{x}^{(h)\prime}_{i;j\pi_h}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j\pi_h}} I_{k,j\pi_h}\theta_{k,j\pi_h}\right]}, \quad h = 2, \cdots, L$$

where I_{π_h} is the *inclusive value* (at level $h+1$) of the branch below node π_h and is defined recursively as follows:

$$I_{\pi_h} = \ln\left\{\sum_{j\in C_{i;\pi_h}} \exp\left[\mathbf{x}^{(h)\prime}_{i;j\pi_h}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j\pi_h}} I_{k,j\pi_h}\theta_{k,j\pi_h}\right]\right\}$$

$$0 \le \theta_{k,\pi_1} \le \cdots \le \theta_{k,\pi_{L-1}}$$

The inclusive value I_{π_h} denotes the average utility that the individual can expect from the branch below π_h. The *dissimilarity parameters* or *inclusive value parameters* ($\theta_{k,j\pi_h}$) are the coefficients of the inclusive values and have values between 0 and 1 if nested logit is the correct model specification. When they all take value 1, the nested logit model is equivalent to the conditional logit model.

At decision level 1, there is no inclusive value, i.e. $I_{\pi_0} = 0$. Therefore, the conditional probability is

$$P_i(j_1|\pi_1) = \frac{\exp\left[\mathbf{x}_{i;j_1\pi_1}^{(1)\prime}\boldsymbol{\beta}^{(1)}\right]}{\sum_{j\in C_{i;\pi_1}}\exp\left[\mathbf{x}_{i;j\pi_1}^{(1)\prime}\boldsymbol{\beta}^{(1)}\right]}$$

The log-likelihood function at level h can then be written

$$\mathcal{L}^{(h)} = \sum_{i=1}^{N}\sum_{\pi_{h'}\in C_{i,\pi_{h+1}}}\sum_{j\in C_{i,\pi_{h'}}} y_{i,j\pi_{h'}}\ln P(C_{i,j\pi_{h'}}|C_{i,\pi_{h'}})$$

where $y_{i,j\pi_{h'}}$ is an indicator variable that has the value of 1 for the selected choice. The full log-likelihood function of the nested logit model is obtained by adding the conditional log-likelihood functions at each level:

$$\mathcal{L} = \sum_{h=1}^{L}\mathcal{L}^{(h)}$$

Note that the log-likelihood functions are computed from conditional probabilities when $h < L$. The nested logit model is estimated using the full information maximum likelihood method.

Decision Tree and Nested Logit

You can view choices as a decision tree and model the decision tree using the nested logit model. You need to use either the NEST statement or the CHOICE= option of the MODEL statement to specify the nested tree structure. Additionally, you need to identify which explanatory variables are used at each level of the decision tree. These explanatory variables are arguments for what is called a *utility function*. The utility function is specified using UTILITY statements. For example, consider a two-level decision tree. The tree structure is displayed in Figure 19.26.

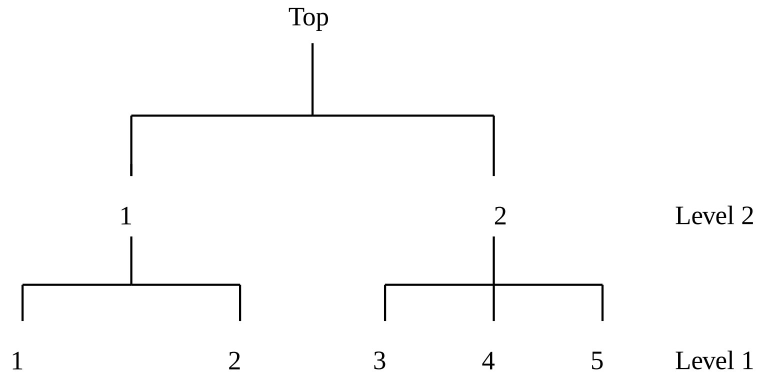


Figure 19.26. Two-Level Decision Tree

A nested logit model with two levels can be specified using the following SAS statements:

```
proc mdc data=one type=nlogit;
   model decision = x1 x2 x3 x4 x5 /
       choice=(upmode 1 2, mode 1 2 3 4 5);
   id pid;
   utility u(1, 3 4 5 @ 2) = x1 x2,
           u(1, 1 2 @ 1) = x3 x4,
           u(2, 1 2) = x5;
run;
```

The DATA=one data set should be arranged as

```
obs  pid  upmode   mode   x1  x2  x3  x4  x5  decision
 1    1     1       1     #   #   #   #   #      1
 2    1     1       2     #   #   #   #   #      0
 3    1     2       3     #   #   #   #   #      0
 4    1     2       4     #   #   #   #   #      0
 5    1     2       5     #   #   #   #   #      0
 6    2     1       1     #   #   #   #   #      0
 7    2     1       2     #   #   #   #   #      0
 8    2     2       3     #   #   #   #   #      0
 9    2     2       4     #   #   #   #   #      0
10    2     2       5     #   #   #   #   #      1
```

All model variables, x1 through x5, are specified in the utility statement. It is required that entries denoted as # have values for model estimation and prediction. The values of the level 2 utility variable x5 should be the same for all the primitive (level 1) alternatives below node 1 at level 2 and, similarly, for all the primitive alternatives below node 2 at level 2. In other words, x5 should have the same value for primitive alternatives 1 and 2 and, similarly, it should have the same value for primitive alternatives 3, 4, and 5. More generally, the values of any level 2 or higher utility function variables should be constant across primitive alternatives under each node for which the utility function applies. Since PROC MDC expects this to be the case, it will only use the values of x5 for the primitive alternatives 1 and 3, ignoring the values for the primitive alternatives 2, 4, and 5. Thus, PROC MDC only uses the values of the utility function variable for the primitive alternatives that come first under each node for which the utility function applies. This behavior applies to any utility function variables that are specified above the first level. The choice variable for level 2 (upmode) should be placed before the first-level choice variable (mode) when the CHOICE= option is given. Alternatively, the NEST statement can be used to specify the decision tree. The following SAS statements fit the same nested logit model:

```
proc mdc data=a type=nlogit;
   model decision = x1 x2 x3 x4 x5 /
       choice=(mode 1 2 3 4 5);
   id pid;
```

```
      utility u(1, 3 4 5 @ 2) = x1 x2,
              u(1, 1 2 @ 1) = x3 x4,
              u(2, 1 2) = x5;
      nest level(1) = (1 2 @ 1, 3 4 5 @ 2),
           level(2) = (1 2 @ 1);
   run;
```

The U(1, 3 4 5 @ 2)= option specifies three choices, 3, 4, and 5, at level 1 of the decision tree. They are connected to the upper branch 2. The specified variables (x1 and x2) are used to model this utility function. The bottom level of the decision tree is level 1. All variables in the UTILITY statement must be included in the MODEL statement. When all choices at the first level share the same variables, you can omit the second argument of the U()= option for that level. However, U(1,) = x1 x2 is not equivalent to

```
   u(1, 3 4 5 @ 2) = x1 x2;
   u(1, 1 2 @ 1) = x1 x2;
```

The CHOICE= variables need to be specified from the top to the bottom level. To forecast demand for new products, stated preference data are widely used. Stated preference data are attractive for market researchers since attribute variations can be controlled. Hensher (1993) explores the advantage of combining revealed preference (market data) and stated preference data. The scale factor (V_{rp}/V_{sp}) can be estimated using the nested logit model with the decision tree structure displayed in Figure 19.27.

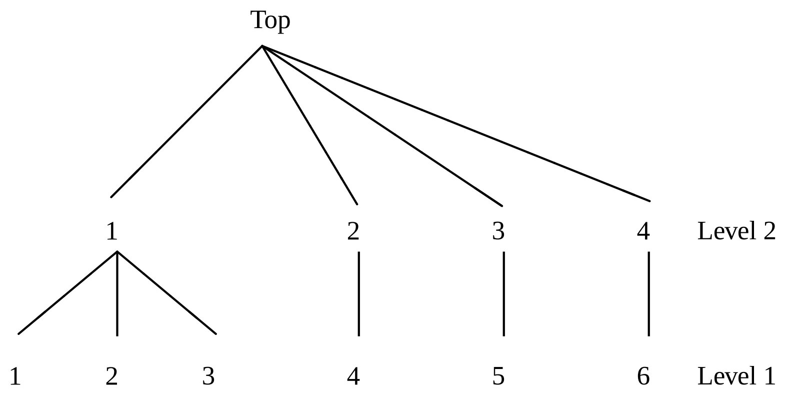


Figure 19.27. Decision Tree for Revealed and Stated Preference Data

Example SAS statements read

```
   proc mdc data=a type=nlogit;
      model decision = x1 x2 x3 / spscale
          choice=(mode 1 2 3 4 5 6);
      id pid;
      utility u(1,) = x1 x2 x3;
      nest level(1) = (1 2 3 @ 1, 4 @ 2, 5 @ 3, 6 @ 4),
           level(2) = (1 2 3 4 @ 1);
   run;
```

The SPSCALE option specifies that parameters of inclusive values for nodes 2, 3, and 4 at level 2 are the same. When you specify the SAMESCALE option, the MDC procedure imposes the same coefficient of inclusive values for choices 1 –4.

Goodness-of-Fit Measures

McFadden (1974) suggests a likelihood ratio index that is analogous to the R^2 in the linear regression model.

$$R_M^2 = 1 - \frac{\ln L}{\ln L_0}$$

where L is the maximum of the log-likelihood function and L_0 is the maximum of the log-likelihood function when all coefficients, except for an intercept term, are zero. McFadden's likelihood ratio index is bounded by 0 and 1.

Estrella (1998) proposes the following requirements for a goodness-of-fit measure to be desirable in discrete choice modeling:

- The measure must take values in $[0, 1]$, where 0 represents no fit and 1 corresponds to perfect fit.
- The measure should be directly related to the valid test statistic for the significance of all slope coefficients.
- The derivative of the measure with respect to the test statistic should comply with corresponding derivatives in a linear regression.

Estrella's measure is written

$$R_{E1}^2 = 1 - \left(\frac{\ln L}{\ln L_0}\right)^{-(2/N)\ln L_0}$$

Estrella suggests an alternative measure

$$R_{E2}^2 = 1 - [(\ln L - K)/\ln L_0]^{-(2/N)\ln L_0}$$

where $\ln L_0$ is computed with null parameter values, N is the number of observations used, and K represents the number of estimated parameters.

Other goodness-of-fit measures are summarized as follows:

$$\begin{aligned}
R_{CU1}^2 &= 1 - \left(\tfrac{L_0}{L}\right)^{\frac{2}{N}} && \text{(Cragg-Uhler 1)} \\
R_{CU2}^2 &= \frac{1-(L_0/L)^{\frac{2}{N}}}{1-L_0^{\frac{2}{N}}} && \text{(Cragg-Uhler 2)} \\
R_A^2 &= \frac{2(\ln L - \ln L_0)}{2(\ln L - \ln L_0)+N} && \text{(Aldrich-Nelson)} \\
R_{VZ}^2 &= R_A^2 \frac{2\ln L_0 - N}{2\ln L_0} && \text{(Veall-Zimmermann)}
\end{aligned}$$

OUTEST= Data Set

The OUTEST= data set contains all the parameters estimated in a MODEL statement. The OUTEST= option can be used when the PROC MDC call contains one MODEL statement. There are additional restrictions. For the HEV and multinomial probit models, you need to specify exactly all possible elements of the choice set, since additional parameters (e.g., SCALE1 or STD1) are generated automatically in the MDC procedure. Therefore, the following SAS statement is not valid when the OUTEST= option is given:

```
proc mdc data=a outest=e;
   model y = x / type=hev choice=(alter);
run;
```

You need to specify all possible choices in the CHOICE= option since the OUTEST= option is specified.

```
proc mdc data=a outest=e;
   model y = x / type=hev choice=(alter 1 2 3);
run;
```

When the NCHOICE= option is given, no additional information on possible choices is required. Therefore, the following is a correct SAS statement:

```
proc mdc data=a outest=e;
   model y = x / type=mprobit nchoice=3;
run;
```

The nested logit model does not produce the OUTEST= data set unless the NEST statement is specified.

Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

DEPVAR	the name of the dependent variable
METHOD	the estimation method
MODEL	the label of the MODEL statement if one is given, or blank otherwise
STATUS	convergence status for optimization
NAME	the name of the row of the covariance matrix for the parameter estimate, if the COVOUT option is specified
LIKL	the log-likelihood value
STDERR	standard error of the parameter estimate, if the COVOUT option is specified
TYPE	PARMS for observations containing parameter estimates, or COV for observations containing covariance matrix elements

The OUTEST= data set contains one observation for the MODEL statement giving the parameter estimates for that model. If the COVOUT option is specified, the OUTEST= data set includes additional observations for the MODEL statement giving the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter associated with that row of the covariance matrix.

ODS Table Names

PROC MDC assigns a name to each table it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 19.2. ODS Tables Produced in PROC MDC

ODS Table Name	Description	Option
	ODS Tables Created by the Model Statement	
FitSummary	Summary of Nonlinear Estimation	default
ResponseProfile	Response Profile	default
GoodnessOfFit	Pseudo-R^2 Measures	default
ParameterEstimates	Parameter Estimates	default
ActiveLinConSol	Linearly Independent Active Linear Constraints	default
CovB	Covariance of Parameter Estimates	COVB
CorrB	Correlation of Parameter Estimates	CORRB

Examples

Example 19.1. Binary Data Modeling

The MDC procedure supports various multinomial choice models. However, binary choice models such as binary logit and probit can also be estimated using PROC MDC since these models are special cases of multinomial models.

Spector and Mazzeo (1980) studied the effectiveness of a new teaching method on students' performance in an economics course. They reported grade point average (gpa), previous knowledge of the material (tuce), a dummy variable for the new teaching method (psi), and the final course grade (grade). grade is recorded as 1 if a student earned the letter grade "A", and 0 otherwise.

The binary logit can be estimated using the conditional logit model. In order to use the MDC procedure, the data are converted so that each possible choice corresponds to one observation.

```
data smdata;
   input gpa tuce psi grade;
   datalines;
2.66      20      0        0
2.89      22      0        0
3.28      24      0        0
2.92      12      0        0
4.00      21      0        1
2.86      17      0        0
2.76      17      0        0
2.87      21      0        0
3.03      25      0        0
3.92      29      0        1
2.63      20      0        0
3.32      23      0        0
3.57      23      0        0
3.26      25      0        1
3.53      26      0        0
2.74      19      0        0
2.75      25      0        0
2.83      19      0        0
3.12      23      1        0
3.16      25      1        1
2.06      22      1        0
3.62      28      1        1
2.89      14      1        0
3.51      26      1        0
3.54      24      1        1
2.83      27      1        1
3.39      17      1        1
2.67      24      1        0
3.65      21      1        1
4.00      23      1        1
3.10      21      1        0
2.39      19      1        1
;

data smdata1;
   set smdata;
   retain id 0;
   id + 1;
   /*-- first choice --*/
   choice1 = 1;
   choice2 = 0;
   decision = (grade = 0);
   gpa_2 = 0;
   tuce_2 = 0;
   psi_2 = 0;
   output;
   /*-- second choice --*/
   choice1 = 0;
   choice2 = 1;
   decision = (grade = 1);
   gpa_2 = gpa;
```

```
      tuce_2 = tuce;
      psi_2 = psi;
      output;
   run;
```

The first 10 observations are displayed in Output 19.1.1. The variables related to grade=0 are omitted since these are not used for binary choice model estimation.

Output 19.1.1. Converted Binary Data

```
          id    decision    choice2    gpa_2    tuce_2    psi_2

           1        1          0        0.00       0        0
           1        0          1        2.66      20        0
           2        1          0        0.00       0        0
           2        0          1        2.89      22        0
           3        1          0        0.00       0        0
           3        0          1        3.28      24        0
           4        1          0        0.00       0        0
           4        0          1        2.92      12        0
           5        0          0        0.00       0        0
           5        1          1        4.00      21        0
```

Consider the choice probability of the conditional logit model for binary choice:

$$P_i(j) = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})}{\sum_{k=1}^{2}\exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}, \quad j = 1, 2$$

The choice probability of the binary logit model is computed based on normalization. The preceding conditional logit model can be converted as

$$P_i(1) = \frac{1}{1 + \exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}$$

$$P_i(2) = \frac{\exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}{1 + \exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}$$

Therefore, you can interpret the binary choice data as the difference between the first and second choice characteristics. In this example, it is assumed that $\mathbf{x}_{i1} = \mathbf{0}$. The binary logit model is estimated and displayed in Output 19.1.2.

```
   proc mdc data=smdata1;
      model decision = choice2 gpa_2 tuce_2 psi_2 /
            type=clogit nchoice=2 covest=hess;
      id id;
   run;
```

Output 19.1.2. Binary Logit Estimates

```
                              The MDC Procedure

                         Conditional Logit Estimates

                             Parameter Estimates

                                              Standard                 Approx
       Parameter      DF      Estimate           Error     t Value    Pr > |t|

       choice2         1      -13.0213          4.9313       -2.64      0.0083
       gpa_2           1        2.8261          1.2629        2.24      0.0252
       tuce_2          1        0.0952          0.1416        0.67      0.5014
       psi_2           1        2.3787          1.0646        2.23      0.0255
```

Consider the choice probability of the multinomial probit model:

$$P_i(j) = P[\epsilon_{i1} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{i1})'\boldsymbol{\beta}, \ldots, \epsilon_{iJ} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{iJ})'\boldsymbol{\beta}]$$

The probabilities of choice of the two alternatives can be written as

$$P_i(1) = P[\epsilon_{i2} - \epsilon_{i1} < (\mathbf{x}_{i1} - \mathbf{x}_{i2})'\boldsymbol{\beta}]$$

$$P_i(2) = P[\epsilon_{i1} - \epsilon_{i2} < (\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta}]$$

where $\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}\right)$. Assume that $\mathbf{x}_{i1} = 0$ and $\sigma_{12} = 0$. The binary probit model is estimated and displayed in Output 19.1.3. You do not get the same estimates as that of the usual binary probit model. The probabilities of choice in the binary probit model are

$$P_i(2) = P[\epsilon_i < \mathbf{x}_i'\boldsymbol{\beta}]$$

$$P_i(1) = 1 - P[\epsilon_i < \mathbf{x}_i'\boldsymbol{\beta}]$$

where $\epsilon_i \sim N(0, 1)$. However, the multinomial probit model has the error variance $\text{Var}(\epsilon_{i2} - \epsilon_{i1}) = \sigma_1^2 + \sigma_2^2$ if ϵ_{i1} and ϵ_{i2} are independent ($\sigma_{12} = 0$). In this example, unit variance restrictions are imposed on choices 1 and 2 ($\sigma_1^2 = \sigma_2^2 = 1$). Therefore, the usual binary probit estimates (and standard errors) can be obtained by multiplying the multinomial probit estimates (and standard errors) in Output 19.1.3 by $1/\sqrt{2}$.

```
proc mdc data=smdata1;
   model decision = choice2 gpa_2 tuce_2 psi_2 /
         type=mprobit nchoice=2 covest=hess
         unitvariance=(1 2);
   id id;
run;
```

Output 19.1.3. Binary Probit Estimates

```
                             The MDC Procedure

                         Multinomial Probit Estimates

                            Parameter Estimates

                                          Standard                Approx
  Parameter      DF      Estimate           Error    t Value    Pr > |t|

  choice2         1      -10.5392          3.5956      -2.93      0.0034
  gpa_2           1        2.2992          0.9813       2.34      0.0191
  tuce_2          1        0.0732          0.1186       0.62      0.5375
  psi_2           1        2.0171          0.8415       2.40      0.0165
```

Example 19.2. Conditional Logit and Data Conversion

In this example, a data preparation macro is introduced. Sometimes, choice-specific information is stored in multiple variables. Since the MDC procedure requires multiple observations for each decision maker, you need to arrange the data so that there is an observation for each subject-alternative (or individual-choice) combination. Simple binary choice data are obtained from Ben-Akiva and Lerman (1985).

```
data travel;
   input auto transit mode $;
   datalines;
52.9   4.4 Transit
4.1   28.5 Transit
4.1   86.9 Auto
56.2  31.6 Transit
51.8  20.2 Transit
0.2   91.2 Auto
27.6  79.7 Auto
89.9  2.2  Transit
41.5  24.5 Transit
95.0  43.5 Transit
99.1  8.4  Transit
18.5  84.0 Auto
82.0  38.0 Auto
 8.6  1.6  Transit
22.5  74.1 Auto
51.4  83.8 Auto
81.0  19.2 Transit
51.0  85.0 Auto
62.2  90.1 Auto
95.1  22.2 Transit
41.6  91.5 Auto
;
run;
```

The travel time is stored in two variables, auto and transit. In addition, the chosen alternatives are stored in a character variable, mode. The choice variable, mode, is converted to a numeric variable, decision, since the MDC procedure only supports numeric variables. The following statements convert the original data set, travel, and estimate the binary logit model. The first 10 observations of a relevant subset of the new data set and the parameter estimates are displayed in Output 19.2.1 and Output 19.2.2, respectively.

```
data new;
   set travel;
   retain id 0;
   id+1;
   /*-- create auto variable --*/
   decision = (upcase(mode) = 'AUTO');
   ttime = auto;
   autodum = 1;
   trandum = 0;
   output;
   /*-- create transit variable --*/
   decision = (upcase(mode) = 'TRANSIT');
   ttime = transit;
   autodum = 0;
   trandum = 1;
   output;
run;

proc print data=new(obs=10);
   var decision autodum trandum ttime;
   id id;
run;

proc mdc data=new;
   model decision = autodum ttime /
         type=clogit nchoice=2;
   id id;
run;
```

Output 19.2.1. Converted Data

```
id    decision    autodum    trandum    ttime

 1        0          1          0        52.9
 1        1          0          1         4.4
 2        0          1          0         4.1
 2        1          0          1        28.5
 3        1          1          0         4.1
 3        0          0          1        86.9
 4        0          1          0        56.2
 4        1          0          1        31.6
 5        0          1          0        51.8
 5        1          0          1        20.2
```

Output 19.2.2. Binary Logit Estimation of Modal Choice Data

```
                                The MDC Procedure

                           Conditional Logit Estimates

                               Parameter Estimates

                                          Standard                  Approx
    Parameter      DF      Estimate          Error    t Value    Pr > |t|

    autodum         1       -0.2376         0.7505      -0.32      0.7516
    ttime           1       -0.0531         0.0206      -2.57      0.0101
```

In order to handle more general cases, you can use the data conversion macro program, %mdcdata. The %mdcdata macro generates choice specific dummy variables and creates multiple observations for each individual. The %mdcdata macro arguments are described below followed by the macro itself.

INDATA=	The name of the input data set.
OUTDATA=	The name of the output data set.
NCHOICE=	Number of alternatives for the choice variable.
ALTSET=	Names the alternatives using characters separated by spaces. There should be a total of nchoice alternatives listed.
DUMLIST=	Names the new dummy variables in the output data set which correspond to the alternatives. Thus, there should be nchoice variables listed here. If not specified, dummy variables will be created as dum_1, ..., dum_J.
VARLIST=	The list of variables from the input data set that will be interleaved in order to reduce the number of variables while expanding the number of observations. This list has a total of NCHOICE times NVAR variables. These variables should be in groups of NCHOICE. For example, if NCHOICE=3, there should be groups of three variables, with each of the three associated with a different alternative.
NVAR=	Number of groups of NCHOICE variables listed in VARLIST=. The %mdcdata macro takes the VARLIST= input variables and reduces them down to only NVAR variables in the output. However, these variables have NCHOICE times as many observations as the input data set variables. The variables are named as x_1, ..., x_n.
SELECT=	Character variable found in the input data set. The values of this variable store the alternatives that were chosen for each subject/individual. The values of this variable must precisely match those listed as ALTSET= values.
ID=	Names the new subject ID variable in the output data set. If not specified, the ID variable will be named id_var.

CHOICE=	Names the new output variable that will identify the alternative value associated with each observation in the output data set. If not specified, the variable will be named alt_var.
DECISION=	Names the new output binary variable that identifies which alternative was chosen. If not specified, the variable will be named dec_var.

```
/*--------------------------------------------------------------
 * name: %mdcdata
 * note: original data contains one observation for each
 *       decision maker (or id).  The choice specific data are
 *       stored in multiple variables.  The created variables are
 *       x_1, ..., x_n.  Choice dummy variables are created as
 *       dum_1, ..., dum_J.  In addition, id_var, alt_var,
 *       and dec_var variables are created.
 * purpose: prepare data for use in the mdc procedure
 *-------------------------------------------------------------*/

%macro mdcdata(indata=,outdata=,varlist=,dumlist=,
               altset=,nchoice=0,nvar=0,select=,
               id=id_var,choice=alt_var,decision=dec_var);

   %if &indata = or &outdata = or &select = %then %do;
      %put ERROR: Not enough information is supplied.;
      %put MDCDATA macro cannot be processed.;
      %goto end_macro;
   %end;

   data &outdata(drop=&varlist _i _j _chk _id _chosen _decide);
      set &indata;
      array _var{&nvar,&nchoice} &varlist;
      array _newvar{&nvar} x_1 - x_%left(&nvar);
      array _dum{&nchoice} %if &dumlist = %then %do;
                              dum_1 - dum_%left(&nchoice)
                           %end;
                           %else %do;
                              &dumlist
                           %end; ;
      array _set{&nchoice} $ _temporary_ (
         %do i = 1 %to &nchoice;
            "%scan(&altset,&i)"
         %end; );
      _chk = 0;
      _id = _n_;
      do _i = 1 to &nchoice;
         %do j = 1 %to &nvar;
            _newvar{&j} = _var{&j,_i};
         %end;
         %if &id = %then %do;
            id_var = _id;
         %end;
         %else %do;
            &id = _id;
         %end;
         %if &choice = %then %do;
            alt_var = _i;
         %end;
         %else %do;
            &choice = _i;
         %end;
         /*-- choice variable is numeric --*/
         %if &altset = %then %do;
            _decide = ( &select = i );
            if ( _decide ) then _chk = _chk + 1;
            %if &decision = %then %do;
               dec_var = _decide;
            %end;
            %else %do;
               &decision = _decide;
            %end;
         %end;
         /*-- choice variable is alphanumeric --*/
         %else %do;
            _chosen = 0;
            do _j = 1 to &nchoice;
               if ( upcase(_set{_j}) = upcase(&select) ) then
                  _chosen = _j;
            end;
            if ( _chosen = 0 ) then
               _decide = 0;
            else _decide = ( _i = _chosen );
            if ( _decide ) then _chk = _chk + 1;
            %if &decision = %then %do;
               dec_var = _decide;
            %end;
            %else %do;
               &decision = _decide;
            %end;
         %end;
         do _j = 1 to &nchoice;
            if ( _i = _j ) then
               _dum{_j} = 1;
            else _dum{_j} = 0;
         end;
         output;
      end;
      /*- check if any decision is not made -*/
      if ( _chk ^= 1 ) then
         put "WARNING: No choices are given for id =" _id;
   run;
   %end_macro:
%mend mdcdata;
```

The following macro invocation shows another way of converting the original data set, travel, for analysis by PROC MDC.

```
%mdcdata(indata=travel,outdata=new,
         varlist=auto transit,
         dumlist=autodum trandum,nchoice=2,nvar=1,
         altset=%str(auto transit),
         select=mode)
```

The first 10 observations of the new data set are displayed in Output 19.2.3.

```
proc print data=new(obs=10);
   var alt_var dec_var autodum trandum x_1;
   id id_var;
run;
```

Output 19.2.3. Converted Data Using %MDCDATA Macro

```
         id_var    alt_var    dec_var    autodum    trandum      x_1

            1         1          0          1          0        52.9
            1         2          1          0          1         4.4
            2         1          0          1          0         4.1
            2         2          1          0          1        28.5
            3         1          1          1          0         4.1
            3         2          0          0          1        86.9
            4         1          0          1          0        56.2
            4         2          1          0          1        31.6
            5         1          0          1          0        51.8
            5         2          1          0          1        20.2
```

The converted explanatory variables are created as x_1, . . ., x_n. In this example, the values of the two variables, auto and transit, are interleaved to produce one variable, x_1, with twice as many observations as the original variables. Additionally, dummy variables (autodum and trandum) listed in the DUMLIST= macro argument are created. When any of the ID=, CHOICE=, or DECISION= macro arguments are not specified, the corresponding variable is automatically produced as id_var, alt_var, or dec_var.

Finally, the binary logit model is estimated. The estimates displayed in Output 19.2.4 are the same as those in Output 19.2.2.

```
proc mdc data=new type=clogit;
   model dec_var = autodum x_1 / nchoice=2;
   id id_var;
run;
```

Output 19.2.4. Conditional Logit Estimates

```
                              The MDC Procedure

                         Conditional Logit Estimates

                             Parameter Estimates

                                         Standard                 Approx
Parameter       DF      Estimate         Error     t Value     Pr > |t|

autodum          1       -0.2376        0.7505       -0.32       0.7516
x_1              1       -0.0531        0.0206       -2.57       0.0101
```

Example 19.3. Correlated Choice Modeling

It is not realistic to assume that the random components of utility for all choices are independent. To analyze correlated data, trinomial choice data (1000 observations) is created using a pseudo-random number generator. The random utility function is

$$U_{ij} = V_{ij} + \epsilon_{ij}, \quad j = 1, 2, 3$$

where

$$\epsilon_{ij} \sim N\left(0, \begin{bmatrix} 2 & .6 & 0 \\ .6 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

```
%let ndim = 3;
%let nobs = 1000;

/*-- generate simulated series --*/
data one;
   array error{&ndim} e1-e3;
   array vtemp{&ndim} _temporary_;
   array lm{6} _temporary_ (1.4142136 0.4242641 0.9055385 0 0 1);
   retain nseed 345678 useed 223344;

   do id = 1 to &nobs;
      index = 0;
      /* generate independent normal variate */
      do i = 1 to &ndim;
         /* index of diagonal element */
         vtemp{i} = rannor(nseed);
      end;
      /* get multivariate normal variate */
      index = 0;
      do i = 1 to &ndim;
         error{i} = 0;
         do j = 1 to i;
            error{i} = error{i} + lm{index+j}*vtemp{j};
         end;
         index = index + i;
      end;
      x1 = 1.0 + 2.0 * ranuni(useed);
      x2 = 1.2 + 2.0 * ranuni(useed);
      x3 = 1.5 + 1.2 * ranuni(useed);
      util1 = 2.0 * x1 + e1;
      util2 = 2.0 * x2 + e2;
      util3 = 2.0 * x3 + e3;
      do i = 1 to &ndim;
         vtemp{i} = 0;
      end;
      if ( util1 > util2 & util1 > util3 ) then
         vtemp{1} = 1;
      else if ( util2 > util1 & util2 > util3 ) then
         vtemp{2} = 1;
      else if ( util3 > util1 & util3 > util2 ) then
         vtemp{3} = 1;
      else continue;
       /*-- first choice --*/
      x = x1;
      mode = 1;
      decision = vtemp{1};
      output;
       /*-- second choice --*/
      x = x2;
      mode = 2;
      decision = vtemp{2};
      output;
       /*-- third choice --*/
      x = x3;
      mode = 3;
      decision = vtemp{3};
      output;
   end;
run;
```

First, the multinomial probit model is estimated. Results show that standard deviation, correlation, and slope estimates are close to the parameter values. Note that $\rho_{12} = \frac{\sigma_{12}}{\sqrt{(\sigma_1^2)(\sigma_2^2)}} = \frac{.6}{\sqrt{(2)(1)}} = .42$, $\sigma_1 = \sqrt{2} = 1.41$, $\sigma_2 = \sqrt{1} = 1$, and the parameter value for the variable x is 2.0. See Output 19.3.1.

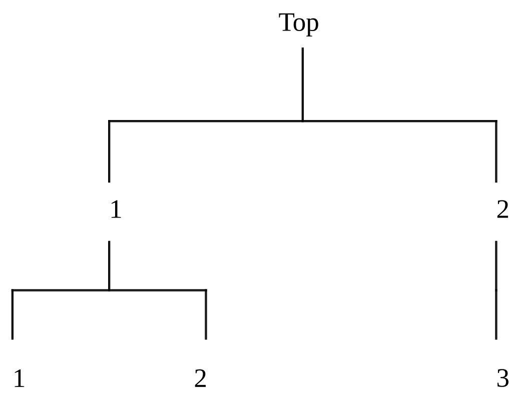
```
   proc mdc data=one randnum=halton nsimul=100;
      model decision = x / type=mprobit
            choice=(mode 1 2 3) covest=op optmethod=qn;
      id id;
   run;
```

Output 19.3.1. Trinomial Probit Model Estimation

```
                                The MDC Procedure

                            Multinomial Probit Estimates

                               Parameter Estimates

                                             Standard                 Approx
        Parameter      DF      Estimate         Error    t Value    Pr > |t|

        x               1        1.7987        0.1202      14.97      <.0001
        STD_1           1        1.2824        0.1468       8.74      <.0001
        RHO_21          1        0.4233        0.1041       4.06      <.0001
```

The nested model is also estimated based on a two-level decision tree. See Output 19.3.2. The estimated result shows that the data supports the nested tree model since the estimates of the inclusive value parameters are significant and are less than 1.

Output 19.3.2. Nested Tree Structure



```
proc mdc data=one;
   model decision = x / type=nlogit
      choice=(mode 1 2 3) covest=op optmethod=qn;
   id id;
   utility u(1,) = x;
   nest level(1) = (1 2 @ 1, 3 @ 2),
        level(2) = (1 2 @ 1);
run;
```

Output 19.3.3. Two-Level Nested Logit

```
                                 The MDC Procedure

                              Nested Logit Estimates

                               Parameter Estimates

                                            Standard                 Approx
     Parameter      DF      Estimate           Error    t Value    Pr > |t|

     x_L1            1        2.6672          0.1978      13.48      <.0001
     INC_L2G1C1      1        0.7911          0.0832       9.51      <.0001
     INC_L2G1C2      1        0.7965          0.0921       8.65      <.0001
```

Example 19.4. Testing for Homoscedasticity of the Utility Function

The “Getting Started” section analyzes an HEV model using Daganzo’s trinomial choice data and displays the HEV parameter estimates in Figure 19.14. The inverted scale estimates for mode “2” and mode “3” suggest that the conditional logit model (which imposes equal variances on random components of utility of all alternatives) might be misleading. The HEV estimation summary from that analysis is repeated in Output 19.4.1.

Output 19.4.1. HEV Estimation Summary ($\theta_1 = 1$)

```
                                 The MDC Procedure

                  Heteroscedastic Extreme Value Model Estimates

                                 Model Fit Summary

                  Dependent Variable                      decision
                  Number of Observations                        50
                  Number of Cases                              150
                  Log Likelihood                         -33.41383
                  Maximum Absolute Gradient              0.0000218
                  Number of Iterations                          11
                  Optimization Method             Dual Quasi-Newton
                  AIC                                     72.82765
                  Schwarz Criterion                       78.56372
```

You can estimate the HEV model with unit scale restrictions on all three alternatives ($\theta_1 = \theta_2 = \theta_3 = 1$) with the following statements. Output 19.4.2 displays the estimation summary.

```
proc mdc data=newdata;
   model decision = ttime / type=hev nchoice=3
         hev=(unitscale=1 2 3, integrate=laguerre) covest=hess;
   id pid;
run;
```

Output 19.4.2. HEV Estimation Summary ($\theta_1 = \theta_2 = \theta_3 = 1$)

```
                                The MDC Procedure

                   Heteroscedastic Extreme Value Model Estimates

                                Model Fit Summary

                   Dependent Variable                        decision
                   Number of Observations                          50
                   Number of Cases                                150
                   Log Likelihood                           -34.12756
                   Maximum Absolute Gradient               6.79511E-9
                   Number of Iterations                             5
                   Optimization Method              Dual Quasi-Newton
                   AIC                                       70.25512
                   Schwarz Criterion                         72.16714
```

The test for scale equivalence (SCALE2=SCALE3=1) is performed using a likelihood ratio test statistic. The following SAS statements compute the test statistic (1.4276) and its p-value (.4898) from the log-likelihood values in Output 19.4.1 and Output 19.4.2:

```
data _null_;
   /*-- test for H0: scale2 = scale3 = 1 --*/
   /*   ln L(max) = -34.1276                */
   /*   ln L(0)   = -33.4138                */
   stat = -2 * ( - 34.1276 + 33.4138 );
   df = 2;
   p_value = 1 - probchi(stat, df);
   put stat p_value;
run;
```

The test statistic fails to reject the null hypothesis of equal scale parameters, which implies that the random utility function is homoscedastic.

A multinomial probit model also allows heteroscedasticity of the random components of utility for different alternatives. Consider the following utility function:

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where

$$\boldsymbol{\epsilon}_i \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}\right)$$

This multinomial probit model is estimated and the estimation summary is displayed in Output 19.4.3.

```
proc mdc data=newdata;
   model decision = ttime / type=mprobit nchoice=3
```

```
            unitvariance=(1 2) covest=hess;
      id pid;
      restrict RHO_31 = 0;
   run;
```

Output 19.4.3. Heteroscedastic Multinomial Probit Estimation Summary

```
                              The MDC Procedure

                        Multinomial Probit Estimates

                              Model Fit Summary

                  Dependent Variable                      decision
                  Number of Observations                        50
                  Number of Cases                              150
                  Log Likelihood                         -33.88604
                  Maximum Absolute Gradient                1.18537
                  Number of Iterations                           8
                  Optimization Method             Dual Quasi-Newton
                  AIC                                     71.77209
                  Schwarz Criterion                       75.59613
```

Next, the multinomial probit model with unit variances ($\sigma_1 = \sigma_2 = \sigma_3 = 1$) is estimated and the estimation summary is displayed in Output 19.4.4.

```
   proc mdc data=newdata;
      model decision = ttime / type=mprobit nchoice=3
            unitvariance=(1 2 3) covest=hess;
      id pid;
   restrict RHO_21 = 0;
   run;
```

Output 19.4.4. Homoscedastic Multinomial Probit Estimation Summary

```
                              The MDC Procedure

                        Multinomial Probit Estimates

                              Model Fit Summary

                  Dependent Variable                      decision
                  Number of Observations                        50
                  Number of Cases                              150
                  Log Likelihood                         -34.54252
                  Maximum Absolute Gradient                2.13340
                  Number of Iterations                           5
                  Optimization Method             Dual Quasi-Newton
                  AIC                                     71.08505
                  Schwarz Criterion                       72.99707
```

The test for homoscedasticity (σ_3 = 1) under $\sigma_1 = \sigma_2 = 1$ shows that the error variance is not heteroscedastic since the test statistic (1.313) is less than $\chi^2_{.05,1} = 3.84$. The marginal probability or p-value computed from the PROBCHI function is $.2519$.

```
data _null_;
   /*-- test for H0: sigma3 = 1 --*/
   /*   ln L(max) = -33.8860      */
   /*   ln L(0)   = -34.5425      */
   stat = -2 * ( -34.5425 + 33.8860 );
   df = 1;
   p_value = 1 - probchi(stat, df);
   put stat p_value;
run;
```

Example 19.5. Choice of Time for Work Trips: Nested Logit Analysis

A sample data of 527 automobile commuters in San Francisco Bay Area has been analyzed by Small (1982) and Brownstone and Small (1989). The regular time of arrival is recorded as between 42.5 minutes early and 17.5 minutes late, and indexed by 12 alternatives using five-minute interval groups. Refer to Small (1982) for more details on this data.

Brownstone and Small (1989) analyzed a two-level nested logit model displayed in Output 19.5.1. The probability of choosing j at level 2 is written

$$P_i(j) = \frac{\exp(\tau_j I_j)}{\sum_{j'=1}^{3} \exp(\tau_{j'} I_{j'})}$$

where $I_{j'}$ is an inclusive value and is computed as

$$I_{j'} = \ln \left[\sum_{k' \in C_{j'}} \exp(\mathbf{x}'_{ik'} \boldsymbol{\beta}) \right]$$

The probability of choosing an alternative k is denoted as

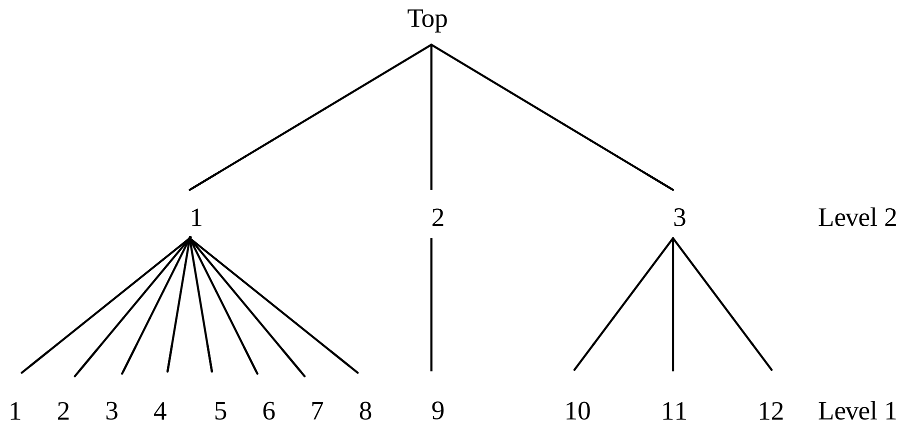
$$P_i(k|j) = \frac{\exp(\mathbf{x}'_{ik} \boldsymbol{\beta})}{\sum_{k' \in C_j} \exp(\mathbf{x}'_{ik'} \boldsymbol{\beta})}$$

The full information maximum likelihood (FIML) method maximizes the following log-likelihood function:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{J} d_{ij} \left[\ln(P_i(k|j)) + \ln(P_i(j)) \right]$$

where $d_{ij} = 1$ if a decision maker i chooses j, and 0 otherwise.

Output 19.5.1. Decision Tree for Two-Level Nested Logit



The following statements estimate the two-level nested logit model.

```
proc mdc data=mylib.small maxit=200 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
      sdl sdlx d2l / type=nlogit choice=(alt);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
run;
```

The estimation summary, discrete response profile, and the FIML estimates are displayed in Output 19.5.2 through Output 19.5.4.

Output 19.5.2. Nested Logit Estimation Summary

```
                         The MDC Procedure

                      Nested Logit Estimates

                         Model Fit Summary

            Dependent Variable                  decision
            Number of Observations                   527
            Number of Cases                         6324
            Log Likelihood                    -990.81912
            Maximum Absolute Gradient         4.93868E-6
            Number of Iterations                      18
            Optimization Method           Newton-Raphson
            AIC                                     2006
            Schwarz Criterion                       2057
```

Output 19.5.3. Discrete Choice Characteristics

```
                              The MDC Procedure

                            Nested Logit Estimates

                          Discrete Response Profile

                      Index       alt      Frequency     Percent

                        0           1              6        1.14
                        1           2             10        1.90
                        2           3             61       11.57
                        3           4             15        2.85
                        4           5             27        5.12
                        5           6             80       15.18
                        6           7             55       10.44
                        7           8             64       12.14
                        8           9            187       35.48
                        9          10             13        2.47
                       10          11              8        1.52
                       11          12              1        0.19
```

Output 19.5.4. Nested Logit Estimates

```
                              The MDC Procedure

                            Nested Logit Estimates

                             Parameter Estimates

                                            Standard                 Approx
        Parameter      DF     Estimate        Error    t Value     Pr > |t|

        r15_L1          1       1.1034       0.1221       9.04       <.0001
        r10_L1          1       0.3931       0.1194       3.29       0.0010
        ttime_L1        1      -0.0465       0.0235      -1.98       0.0474
        ttime_cp_L1     1      -0.0498       0.0305      -1.63       0.1028
        sde_L1          1      -0.6618       0.0833      -7.95       <.0001
        sde_cp_L1       1       0.0519       0.1278       0.41       0.6850
        sdl_L1          1      -2.1006       0.5062      -4.15       <.0001
        sdlx_L1         1      -3.5240       1.5346      -2.30       0.0217
        d2l_L1          1      -1.0941       0.3273      -3.34       0.0008
        INC_L2G1C1      1       0.6762       0.2754       2.46       0.0141
        INC_L2G1C2      1       1.0906       0.3090       3.53       0.0004
        INC_L2G1C3      1       0.7622       0.1649       4.62       <.0001
```

Brownstone and Small (1989) also estimate the two-level nested logit model with equal scale parameter constraints, $\tau_1 = \tau_2 = \tau_3$. Replication of their model results is displayed in Output 19.5.5 and Output 19.5.6. The parameter estimates and standard errors are almost identical to those in Brownstone and Small (1989, p. 69).

```
   proc mdc data=mylib.small maxit=200 outest=a;
      model decision = r15 r10 ttime ttime_cp sde sde_cp
                       sdl sdlx d2l / samescale
            type=nlogit choice=(alt);
      id id;
      utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                       sdl sdlx d2l;
      nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
           level(2) = (1 2 3 @ 1);
   run;
```

Output 19.5.5. Nested Logit Estimation Summary with Equal Dissimilarity Parameters

```
                               The MDC Procedure

                             Nested Logit Estimates

                               Model Fit Summary

                  Dependent Variable                 decision
                  Number of Observations                  527
                  Number of Cases                        6324
                  Log Likelihood                   -994.39402
                  Maximum Absolute Gradient        2.97172E-6
                  Number of Iterations                     16
                  Optimization Method          Newton-Raphson
                  AIC                                    2009
                  Schwarz Criterion                      2051
```

Output 19.5.6. Nested Logit Estimates with Equal Dissimilarity Parameters

```
                               The MDC Procedure

                             Nested Logit Estimates

                              Parameter Estimates

                                           Standard                 Approx
       Parameter       DF      Estimate       Error     t Value    Pr > |t|

       r15_L1           1        1.1345      0.1092       10.39      <.0001
       r10_L1           1        0.4194      0.1081        3.88      0.0001
       ttime_L1         1       -0.1626      0.0609       -2.67      0.0076
       ttime_cp_L1      1        0.1285      0.0853        1.51      0.1319
       sde_L1           1       -0.7548      0.0669      -11.28      <.0001
       sde_cp_L1        1        0.2292      0.0981        2.34      0.0195
       sdl_L1           1       -2.0719      0.4860       -4.26      <.0001
       sdlx_L1          1       -2.8216      1.2560       -2.25      0.0247
       d2l_L1           1       -1.3164      0.3474       -3.79      0.0002
       INC_L2G1         1        0.8059      0.1705        4.73      <.0001
```

However, the test statistic for $H_0 : \tau_1 = \tau_2 = \tau_3$ rejects the null hypothesis at the 5% significance level since $-2 * (\ln L(0) - \ln L) = 7.15 > \chi^2_{.05,2} = 5.99$. The p-value is computed as $.0280$.

```
   data _null_;
      /*-- test for H0: tau1 = tau2 = tau3 --*/
      /*   ln L(max) = -990.8191             */
      /*   ln L(0)   = -994.3940             */
      stat = -2 * ( -994.3940 + 990.8191 );
      df = 2;
      p_value = 1 - probchi(stat, df);
      put stat p_value;
   run;
```

Acknowledgments

Professor Kenneth Small has provided the work trip data that is used in the "Examples" section. This data was collected for the urban travel demand forecasting project, which was carried out by McFadden et al. (1977). The project was supported by the National Science Foundation, Research Applied to National Needs Program through grants GI-43740 and APR74-20392, and the Alfred P. Sloan Foundation, through grant 74-21-8.

References

Abramowitz, M. and Stegun, A. (1970), *Handbook of Mathematical Functions*, New York: Dover Press.

Amemiya, T. (1981), "Qualitative Response Models: A Survey," *Journal of Economic Literature*, 19, 483–536.

Amemiya, T. (1985), *Advanced Econometrics*, Cambridge: Harvard University Press.

Ben-Akiva, M. and Lerman, S.R. (1985), *Discrete Choice Analysis*, Cambridge: MIT Press.

Bhat, C.R. (1995), "A Heteroscedastic Extreme Value Model of Intercity Travel Mode Choice," *Transportation Research*, 29, 471–483.

Brownstone, D. and Small, K.A. (1989), "Efficient Estimation of Nested Logit Models," *Journal of Business and Statistics*, 7, 67–74.

Brownstone, D. and Train, K. (1999), "Forecasting New Product Penetration with Flexible Substitution Patterns," *Journal of Econometrics*, 89, 109–129.

Daganzo, C. (1979), *Multinomial Probit: The Theory and Its Application to Demand Forecasting*, New York: Academic Press.

Daganzo, C. and Kusnic, M. (1993), "Two Properties of the Nested Logit Model," *Transportation Science*, 27, 395–400.

Estrella, A. (1998), "A New Measure of Fit for Equations With Dichotomous Dependent Variables," *Journal of Business and Economic Statistics*, 16, 198–205.

Geweke, J. (1989),"Bayesian Inference in Econometric Models Using Monte Carlo Integration," *Econometrica*, 57, 1317–1340.

Geweke, J., Keane, M., and Runkle, D. (1994), "Alternative Computational Approaches to Inference in the Multinomial Probit Model," *Review of Economics and Statistics*, 76, 609–632.

Godfrey, L.G. (1988), *Misspecification Tests in Econometrics*, Cambridge: Cambridge University Press.

Green, W.H. (1997), *Econometric Analysis*, Upper Saddle River, N.J.: Prentice Hall.

Hajivassiliou, V.A. (1993), "Simulation Estimation Methods for Limited Dependent Variable Models," in *Handbook of Statistics*, vol. 11, ed. G.S. Maddala, C.R. Rao, and H.D. Vinod, New York: Elsevier Science Publishing.

Hajivassiliou, V., McFadden, D., and Ruud, P. (1996), "Simulation of Multivariate Normal Rectangle Probabilities and Their Derivatives: Theoretical and Computational Results," *Journal of Econometrics*, 72, 85–134.

Hensher, D.A. (1986), "Sequential and Full Information Maximum Likelihood Estimation of a Nested Logit Model," *Review of Economics and Statistics*, 68, 657–667.

Hensher, D.A. (1993), "Using Stated Response Choice Data to Enrich Revealed Preference Discrete Choice Models," *Marketing Letters*, 4, 139–151.

Keane, M.P. (1994), "A Computationally Practical Simulation Estimator for Panel Data," *Econometrica*, 62, 95–116.

Keane, M.P. (1997), "Current Issues in Discrete Choice Modeling," *Marketing Letters*, 8, 307–322.

LaMotte, L.R. (1994), "A Note on the Role of Independence in t Statistics Constructed from Linear Statistics in Regression Models," *The American Statistician*, 48, 238–240.

Luce, R.D. (1959), *Individual Choice Behavior*: *A Theoretical Analysis*, New York: Wiley.

Maddala, G.S. (1983), *Limited Dependent and Qualitative Variables in Econometrics*, Cambridge: Cambridge University Press.

McFadden, D. (1974), "Conditional Logit Analysis of Qualitative Choice Behavior," in *Frontiers in Econometrics*, ed. P. Zarembka, New York: Academic Press.

McFadden, D. (1978), "Modelling the Choice of Residential Location," in *Spatial Interaction Theory and Planning Models*, ed. A. Karlqvist, L. Lundqvist, F. Snickars, and J. Weibull, Amsterdam: North Holland.

McFadden, D. (1981), "Econometric Models of Probabilistic Choice," in *Structural Analysis of Discrete Data with Econometric Applications*, ed. C.F. Manski and D. McFadden, Cambridge: MIT Press.

McFadden, D. and Ruud, P.A. (1994), "Estimation by Simulation," *Review of Economics and Statistics*, 76, 591–608.

McFadden, D., Talvitie, A.P., and Associates (1977), *The Urban Travel Demand Forecasting Project: Phase I Final Report Series*, vol. 5, The Institute of Transportation Studies, University of California, Berkeley.

Powers, D.A. and Xie, Y. (2000), *Statistical Methods for Categorical Data Analysis*, San Diego: Academic Press.

Schmidt, P. and Strauss, R. (1975), "The Prediction of Occupation Using Multiple Logit Models," *International Economic Review*, 16, 471–486.

Small, K. (1982), "The Scheduling of Consumer Activities: Work Trips," *American Economic Review*, 72, 467–479.

Spector, L. and Mazzeo, M. (1980), “Probit Analysis and Economic Education,” *Journal of Economic Education*, 11, 37–44.

Swait, J. and Bernardino, A. (2000), “Distingushing Taste Variation from Error Structure in Discrete Choice Data,” *Transportation Research Part B*, 34, 1–15.

Theil, H. (1969), “A Multinomial Extension of the Linear Logit Model,” *International Economic Review*, 10, 251–259.

Train, K.E., Ben-Akiva, M., and Atherton, T. (1989), “Consumption Patterns and Self-Selecting Tariffs,” *Review of Economics and Statistics*, 71, 62–73.

Chapter 20
The MODEL Procedure

Chapter Contents

Chapter 20
The MODEL Procedure

Overview

The MODEL procedure analyzes models in which the relationships among the variables comprise a system of one or more nonlinear equations. Primary uses of the MODEL procedure are estimation, simulation, and forecasting of nonlinear simultaneous equation models.

PROC MODEL features include

- SAS programming statements to define simultaneous systems of nonlinear equations
- tools to analyze the structure of the simultaneous equation system
- ARIMA, PDL, and other dynamic modeling capabilities
- tools to specify and estimate the error covariance structure
- tools to estimate and solve ordinary differential equations
- the following methods for parameter estimation:
  - Ordinary Least Squares (OLS)
  - Two-Stage Least Squares (2SLS)
  - Seemingly Unrelated Regression (SUR) and iterative SUR (ITSUR)
  - Three-Stage Least Squares (3SLS) and iterative 3SLS (IT3SLS)
  - Generalized Method of Moments (GMM)
  - Simulated Method of Moments (SMM)
  - Full Information Maximum Likelihood (FIML)
  - General Log-Likelihood maximization
- simulation and forecasting capabilities
- Monte Carlo simulation
- goal seeking solutions

Experimental graphics are now available with the MODEL procedure. For more information, see the "ODS Graphics" section on page 1166.

A system of equations can be nonlinear in the parameters, nonlinear in the observed variables, or nonlinear in both the parameters and the variables. *Nonlinear* in the parameters means that the mathematical relationship between the variables and parameters is not required to have a linear form. (A linear model is a special case of a nonlinear model.) A general nonlinear system of equations can be written as

$$\begin{aligned}
q_1(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{1,t} \\
q_2(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{2,t} \\
&\vdots \\
q_g(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{g,t}
\end{aligned}$$

where $y_{i,t}$ is an endogenous variable, $x_{i,t}$ is an exogenous variable, θ_i is a parameter, and ϵ_i is the unknown error. The subscript t represents time or some index to the data. In econometrics literature, the observed variables are either *endogenous* (dependent) variables or *exogenous* (independent) variables. This system can be written more succinctly in vector form as

$$\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) = \epsilon_t$$

This system of equations is in *general form* because the error term is by itself on one side of the equality. Systems can also be written in *normalized form* by placing the endogenous variable on one side of the equality, with each equation defining a predicted value for a unique endogenous variable. A normalized form equation system can be written in vector notation as

$$\mathbf{y}_t = \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \theta) + \epsilon_t.$$

PROC MODEL handles equations written in both forms.

Econometric models often explain the current values of the endogenous variables as functions of past values of exogenous and endogenous variables. These past values are referred to as *lagged* values, and the variable x_{t-i} is called lag *i* of the variable x_t. Using lagged variables, you can create a *dynamic*, or time dependent, model. In the preceding model systems, the lagged exogenous and endogenous variables are included as part of the exogenous variables.

If the data are time series, so that *t* indexes time (see Chapter 2, “Working with Time Series Data,” for more information on time series), it is possible that ϵ_t depends on ϵ_{t-i} or, more generally, the ϵ_t’s are not identically and independently distributed. If the errors of a model system are autocorrelated, the standard error of the estimates of the parameters of the system will be inflated.

Sometimes the ϵ_i’s are not identically distributed because the variance of ϵ is not constant. This is known as *heteroscedasticity*. Heteroscedasticity in an estimated model can also inflate the standard error of the estimates of the parameters. Using a weighted estimation can sometimes eliminate this problem. Alternately, a variance model such as GARCH or EGARCH can be estimated to correct for heteroscedasticity. If the proper weighting scheme and the form of the error model is difficult to determine, generalized methods of moments (GMM) estimation can be used to determine parameter estimates with very few assumptions about the form of the error process.

Other problems may also arise when estimating systems of equations. Consider the system of equations:

$$\begin{aligned} y_{1,t} &= \theta_1 + (\theta_2 + \theta_3\theta_4^t)^{-1} + \theta_5 y_{2,t} + \epsilon_{1,t} \\ y_{2,t} &= \theta_6 + (\theta_7 + \theta_8\theta_9^t)^{-1} + \theta_{10} y_{1,t} + \epsilon_{2,t} \end{aligned}$$

which is nonlinear in its parameters and cannot be estimated with linear regression. This system of equations represents a rudimentary predator-prey process with y_1 as the prey and y_2 as the predator (the second term in both equations is a logistics curve). The two equations must be estimated simultaneously because of the cross dependency of y's. This cross-dependency makes ϵ_1 and ϵ_2 violate the assumption of indpendence. Nonlinear ordinary least-squares estimation of these equations will produce biased and inconsistent parameter estimates. This is called *simultaneous equation bias*.

One method to remove simultaneous equation bias, in the linear case, is to replace the endogenous variables on the right-hand side of the equations with predicted values that are uncorrelated with the error terms. These predicted values can be obtained through a preliminary, or "first stage," *instrumental variable regression*. *Instrumental variables*, which are uncorrelated with the error term, are used as regressors to model the predicted values. The parameter estimates are obtained by a second regression using the predicted values of the regressors. This process is called *two-stage least squares*.

In the nonlinear case, nonlinear ordinary least-squares estimation is performed iteratively using a linearization of the model with respect to the parameters. The instrumental solution to simultaneous equation bias in the nonlinear case is the same as the linear case except the linearization of the model with respect to the parameters is predicted by the instrumental regression. Nonlinear two-stage least squares is one of several instrumental variables methods available in the MODEL procedure to handle simultaneous equation bias.

When you have a system of several regression equations, the random errors of the equations can be correlated. In this case, the large-sample efficiency of the estimation can be improved by using a joint generalized least-squares method that takes the cross-equation correlations into account. If the equations are not simultaneous (no dependent regressors), then *seemingly unrelated regression* (SUR) can be used. The SUR method requires an estimate of the cross-equation error covariance matrix, Σ. The usual approach is to first fit the equations using OLS, compute an estimate $\hat{\Sigma}$ from the OLS residuals, and then perform the SUR estimation based on $\hat{\Sigma}$. The MODEL procedure estimates Σ by default, or you can supply your own estimate of Σ.

If the equation system is simultaneous, you can combine the 2SLS and SUR methods to take into account both simultaneous equation bias and cross-equation correlation of the errors. This is called *three-stage least squares* or 3SLS.

A different approach to the simultaneous equation bias problem is the full information maximum likelihood, or FIML, estimation method. FIML does not require instrumental variables, but it assumes that the equation errors have a multivariate normal distribution. 2SLS and 3SLS estimation do not assume a particular distribution for the errors.

Other non-normal error distribution models can be estimated as well. The centered *t*-distribution with estimated degrees of freedom and non-constant variance is an additional built in likelihood function. If the distribution of the equation errors is not normal or *t* but known, then the log likelihood can be specified using the ERRORMODEL statement.

Once a nonlinear model has been estimated, it can be used to obtain forecasts. If the model is linear in the variables you want to forecast, a simple linear solve can generate the forecasts. If the system is nonlinear, an iterative procedure must be used. The preceding example system is linear in its endogenous variables. The MODEL procedure's SOLVE statement is used to forecast nonlinear models.

One of the main purposes of creating models is to obtain an understanding of the relationship among the variables. There are usually only a few variables in a model you can control (for example, the amount of money spent on advertising). Often you want to determine how to change the variables under your control to obtain some target goal. This process is called *goal seeking*. PROC MODEL allows you to solve for any subset of the variables in a system of equations given values for the remaining variables.

The nonlinearity of a model creates two problems with the forecasts: the forecast errors are not normally distributed with zero mean, and no formula exits to calculate the forecast confidence intervals. PROC MODEL provides Monte Carlo techniques, which, when used with the covariance of the parameters and error covariance matrix, can produce approximate error bounds on the forecasts. The following distributions on the errors are supported for multivariate Monte Carlo simulation:

- Cauchy
- Chi-squared
- Empirical
- *F*
- Poisson
- *t*
- Uniform

A transformation technique is used to create a covariance matrix for generating the correct innovations in a Monte Carlo simulation.

Getting Started

This section introduces the MODEL procedure and shows how to use PROC MODEL for several kinds of nonlinear regression analysis and nonlinear systems simulation problems.

Nonlinear Regression Analysis

One of the most important uses of PROC MODEL is to estimate unknown parameters in a nonlinear model. A simple nonlinear model has the form:

$$y = f(\mathbf{x}, \theta) + \epsilon$$

where $\mathbf{x}$ is a vector of exogenous variables. To estimate unknown parameters using PROC MODEL, do the following:

1. Use the DATA= option in a PROC MODEL statement to specify the input SAS data set containing $\mathbf{y}$ and $\mathbf{x}$, the observed values of the variables.
2. Write the equation for the model using SAS programming statements, including all parameters and arithmetic operators but leaving off the unobserved error component, ϵ.
3. Use a FIT statement to fit the model equation to the input data to determine the unknown parameters, θ.

An Example

The SASHELP library contains the data set CITIMON, which contains the variable LHUR, the monthly unemployment figures, and the variable IP, the monthly industrial production index. You suspect that the unemployment rates are inversely proportional to the industrial production index. Assume that these variables are related by the following nonlinear equation:

$$lhur = \frac{1}{a \cdot \mathrm{ip} + b} + c + \epsilon$$

In this equation *a*, *b*, and *c* are unknown coefficients and ϵ is an unobserved random error.

The following statements illustrate how to use PROC MODEL to estimate values for *a*, *b*, and *c* from the data in SASHELP.CITIMON.

```
proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
run;
```

Notice that the model equation is written as a SAS assignment statement. The variable LHUR is assumed to be the dependent variable because it is named in the FIT statement and is on the left-hand side of the assignment.

PROC MODEL determines that LHUR and IP are observed variables because they are in the input data set. A, B, and C are treated as unknown parameters to be estimated from the data because they are not in the input data set. If the data set contained a variable named A, B, or C, you would need to explicitly declare the parameters with a PARMS statement.

In response to the FIT statement, PROC MODEL estimates values for A, B, and C using nonlinear least squares and prints the results. The first part of the output is a “Model Summary table, shown in Figure 20.1.

```
                              The MODEL Procedure

                                 Model Summary

                           Model Variables         1
                           Parameters              3
                           Equations               1
                           Number of Statements    1


                             Model Variables  LHUR
                                  Parameters  a b c
                                   Equations  LHUR
```

Figure 20.1. Model Summary Report

This table details the size of the model, including the number of programming statements defining the model, and lists the dependent variables (LHUR in this case), the unknown parameters (A, B, and C), and the model equations. In this case the equation is named for the dependent variable, LHUR.

PROC MODEL then prints a summary of the estimation problem, as shown in Figure 20.2.

```
                              The MODEL Procedure

                          The Equation to Estimate is

                             LHUR =  F(a, b, c(1))
```

Figure 20.2. Estimation Problem Report

The notation used in the summary of the estimation problem indicates that LHUR is a function of A, B, and C, which are to be estimated by fitting the function to the data. If the partial derivative of the equation with respect to a parameter is a simple variable or constant, the derivative is shown in parentheses after the parameter name. In this case, the derivative with respect to the intercept C is 1. The derivatives with respect to A and B are complex expressions and so are not shown.

Next, PROC MODEL prints an estimation summary as shown in Figure 20.3.

```
                         The MODEL Procedure
                        OLS Estimation Summary

                           Data Set Options

                       DATA=    SASHELP.CITIMON


                         Minimization Summary

                   Parameters Estimated           3
                   Method                     Gauss
                   Iterations                    10


                      Final Convergence Criteria

                      R                  0.000737
                      PPC(b)             0.003943
                      RPC(b)              0.00968
                      Object             4.784E-6
                      Trace(S)           0.533325
                      Objective Value    0.522214


                         Observations Processed

                             Read       145
                             Solved     145
                             Used       144
                             Missing      1
```

Figure 20.3. Estimation Summary Report

The estimation summary provides information on the iterative process used to compute the estimates. The heading "OLS Estimation Summary" indicates that the nonlinear ordinary least-squares (OLS) estimation method is used. This table indicates that all 3 parameters were estimated successfully using 144 nonmissing observations from the data set SASHELP.CITIMON. Calculating the estimates required 10 iterations of the GAUSS method. Various measures of how well the iterative process converged are also shown. For example, the "RPC(B)" value 0.00968 means that on the final iteration the largest relative change in any estimate was for parameter B, which changed by .968 percent. See the section "Convergence Criteria" later in this chapter for details.

PROC MODEL then prints the estimation results. The first part of this table is the summary of residual errors, shown in Figure 20.4.

```
                         The MODEL Procedure

              Nonlinear OLS Summary of Residual Errors

                    DF      DF                                          Adj
Equation         Model   Error        SSE        MSE    R-Square       R-Sq

LHUR                 3     141    75.1989     0.5333      0.7472     0.7436
```

Figure 20.4. Summary of Residual Errors Report

This table lists the sum of squared errors (SSE), the mean square error (MSE), the root mean square error (Root MSE), and the R^2 and adjusted R^2 statistics. The R^2 value of .7472 means that the estimated model explains approximately 75 percent more of the variability in LHUR than a mean model explains.

Following the summary of residual errors is the parameter estimates table, shown in Figure 20.5.

```
                              The MODEL Procedure

                      Nonlinear OLS Parameter Estimates

                                    Approx                      Approx
          Parameter       Estimate     Std Err    t Value     Pr > |t|

          a               0.009046     0.00343       2.63       0.0094
          b               -0.57059      0.2617      -2.18       0.0309
          c               3.337151      0.7297       4.57       <.0001
```

Figure 20.5. Parameter Estimates

Because the model is nonlinear, the standard error of the estimate, the t value, and its significance level are only approximate. These values are computed using asymptotic formulas that are correct for large sample sizes but only approximately correct for smaller samples. Thus, you should use caution in interpreting these statistics for nonlinear models, especially for small sample sizes. For linear models, these results are exact and are the same as standard linear regression.

The last part of the output produced by the FIT statement is shown in Figure 20.6.

```
                              The MODEL Procedure

               Number of Observations     Statistics for System

               Used               144    Objective          0.5222
               Missing              1    Objective*N       75.1989
```

Figure 20.6. System Summary Statistics

This table lists the objective value for the estimation of the nonlinear system, which is a weighted system mean square error. This statistic can be used for testing cross-equation restrictions in multi-equation regression problems. See the section “Restrictions and Bounds on Parameters” for details. Since there is only a single equation in this case, the objective value is the same as the residual MSE for LHUR except that the objective value does not include a degrees of freedom correction. This can be seen in the fact that “Objective*N” equals the residual SSE, 75.1989. N is 144, the number of observations used.

Convergence and Starting Values

Computing parameter estimates for nonlinear equations requires an iterative process. Starting with an initial guess for the parameter values, PROC MODEL tries different parameter values until the objective function of the estimation method is minimized.

(The objective function of the estimation method is sometimes called the *fitting function*.) This process does not always succeed, and whether it does succeed depends greatly on the starting values used. By default, PROC MODEL uses the starting value .0001 for all parameters.

Consequently, in order to use PROC MODEL to achieve convergence of parameter estimates, you need to know two things: how to recognize convergence failure by interpreting diagnostic output, and how to specify reasonable starting values. The MODEL procedure includes alternate iterative techniques and grid search capabilities to aid in finding estimates. See the section "Troubleshooting Convergence Problems" for more details.

Nonlinear Systems Regression

If a model has more than one endogenous variable, several facts need to be considered in the choice of an estimation method. If the model has endogenous regressors, then an instrumental variables method such as 2SLS or 3SLS can be used to avoid simultaneous equation bias. Instrumental variables must be provided to use these methods. A discussion of possible choices for instrumental variables is provided in the the section "Choice of Instruments" on page 1135 in this chapter.

The following is an example of the use of 2SLS and the INSTRUMENTS statement:

```
proc model data=test2 ;
   exogenous x1 x2;
   parms a1 a2 b2 2.5 c2 55 d1;

   y1 = a1 * y2 + b2 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 2sls;
   instruments b2 c2 _exog_;
run;
```

The estimation method selected is added after the slash (/) on the FIT statement. The INSTRUMENTS statement follows the FIT statement and in this case selects all the exogenous variables as instruments with the _EXOG_ keyword. The parameters B2 and C2 on the instruments list request that the derivatives with respect to B2 and C2 be additional instruments.

Full information maximum likelihood (FIML) can also be used to avoid simultaneous equation bias. FIML is computationally more expensive than an instrumental variables method and assumes that the errors are normally distributed. On the other hand, FIML does not require the specification of instruments. FIML is selected with the FIML option on the FIT statement.

The preceding example is estimated with FIML using the following statements:

```
proc model data=test2 ;
   exogenous x1 x2;
   parms a1 a2 b2 2.5 c2 55 d1;

   y1 = a1 * y2 + b2 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / fiml;
run;
```

General Form Models

The single equation example shown in the preceding section was written in normalized form and specified as an assignment of the regression function to the dependent variable LHUR. However, sometimes it is impossible or inconvenient to write a nonlinear model in normalized form.

To write a general form equation, give the equation a name with the prefix “EQ.” . This EQ.-prefixed variable represents the equation error. Write the equation as an assignment to this variable.

For example, suppose you have the following nonlinear model relating the variables *x* and *y*:

$$\epsilon = a + b\ln(cy + dx)$$

Naming this equation ‘one’, you can fit this model with the following statements:

```
proc model data=xydata;
   eq.one = a + b * log( c * y + d * x );
   fit one;
run;
```

The use of the EQ. prefix tells PROC MODEL that the variable is an error term and that it should not expect actual values for the variable ONE in the input data set.

Supply and Demand Models

General form specifications are often useful when you have several equations for the same dependent variable. This is common in supply and demand models, where both the supply equation and the demand equation are written as predictions for quantity as functions of price.

For example, consider the following supply and demand system:

$$\text{(supply)} \qquad \text{quantity} = \alpha_1 + \alpha_2\ \text{price} + \epsilon_1$$

$$\text{(demand)} \qquad \text{quantity} = \beta_1 + \beta_2 \text{ price} + \beta_3 \text{ income} + \epsilon_2$$

Assume the *quantity* of interest is the amount of energy consumed in the U.S.; the *price* is the price of gasoline, and the *income* variable is the consumer debt. When the market is at equilibrium, these equations determine the market price and the equilibrium quantity. These equations are written in general form as

$$\epsilon_1 = quantity - (\alpha_1 + \alpha_2\, price)$$

$$\epsilon_2 = quantity - (\beta_1 + \beta_2\, price + \beta_3\, income)$$

Note that the endogenous variables *quantity* and *price* depend on two error terms so that OLS should not be used. The following example uses three-stage least-squares estimation.

Data for this model is obtained from the SASHELP.CITIMON data set.

```
title1 'Supply-Demand Model using General-form Equations';
proc model data=sashelp.citimon;
   endogenous eegp eec;
   exogenous exvus cciutc;
   parameters a1 a2 b1 b2 b3 ;
   label eegp   = 'Gasoline Retail Price'
         eec    = 'Energy Consumption'
         cciutc = 'Consumer Debt';

   /* -------- Supply equation ------------- */
   eq.supply = eec - (a1 + a2 * eegp );

   /* -------- Demand equation ------------- */
   eq.demand = eec - (b1 + b2 * eegp + b3 * cciutc);

   /* -------- Instrumental variables -------*/
   lageegp = lag(eegp); lag2eegp=lag2(eegp);

   /* -------- Estimate parameters --------- */
   fit supply demand / n3sls fsrsq;
   instruments _EXOG_ lageegp lag2eegp;
run;
```

The FIT statement specifies the two equations to estimate and the method of estimation, N3SLS. Note that '3SLS' is an alias for N3SLS. The option FSRSQ is selected to get a report of the first stage R^2 to determine the acceptability of the selected instruments.

Since three-stage least squares is an instrumental variables method, instruments are specified with the INSTRUMENTS statement. The instruments selected are all the exogenous variables, selected with the _EXOG_ option, and two lags of the variable EEGP, LAGEEGP and LAG2EEGP.

The data set CITIMON has four observations that generate missing values because values for either EEGP, EEC, or CCIUTC are missing. This is revealed in the "Observations Processed" output shown in Figure 20.7. Missing values are also generated when the equations cannot be computed for a given observation. Missing observations are not used in the estimation.

```
                Supply-Demand Model using General-form Equations

                              The MODEL Procedure
                            3SLS Estimation Summary

                             Observations Processed

                                Read        145
                                Solved      143
                                First         3
                                Last        145
                                Used        139
                                Missing       4
                                Lagged        2
```

Figure 20.7. Supply-Demand Observations Processed

The lags used to create the instruments also reduce the number of observations used. In this case, the first 2 observations were used to fill the lags of EEGP.

The data set has a total of 145 observations, of which 4 generated missing values and 2 were used to fill lags, which left 139 observations for the estimation. In the estimation summary, in Figure 20.8, the total degrees of freedom for the model and error is 139.

```
                Supply-Demand Model using General-form Equations

                              The MODEL Procedure

                   Nonlinear 3SLS Summary of Residual Errors

                  DF     DF                                                  Adj
Equation       Model  Error        SSE        MSE   Root MSE  R-Square      R-Sq

supply             2    137    43.2677     0.3158     0.5620
demand             3    136    39.5791     0.2910     0.5395


                        Nonlinear 3SLS Parameter Estimates

                                                                          1st
                                  Approx                   Approx       Stage
  Parameter        Estimate      Std Err    t Value     Pr > |t|     R-Square

  a1                7.30952       0.3799      19.24       <.0001       1.0000
  a2               -0.00853      0.00328      -2.60       0.0103       0.9617
  b1                6.82196       0.3788      18.01       <.0001       1.0000
  b2               -0.00614      0.00303      -2.02       0.0450       0.9617
  b3                   9E-7     3.165E-7       2.84       0.0051       1.0000
```

Figure 20.8. Supply-Demand Parameter Estimates

One disadvantage of specifying equations in general form is that there are no actual values associated with the equation, so the R^2 statistic cannot be computed.

Solving Simultaneous Nonlinear Equation Systems

You can use a SOLVE statement to solve the nonlinear equation system for some variables when the values of other variables are given.

Consider the demand and supply model shown in the preceding example. The following statement computes equilibrium price (EEGP) and quantity (EEC) values for given observed cost (CCIUTC) values and stores them in the output data set EQUILIB.

```
title1 'Supply-Demand Model using General-form Equations';
proc model data=sashelp.citimon;
   endogenous eegp eec;
   exogenous exvus cciutc;
   parameters a1 a2 a3 b1 b2 ;
   label eegp   = 'Gasoline Retail Price'
         eec    = 'Energy Consumption'
         cciutc = 'Consumer Debt';

   /* -------- Supply equation ------------- */
   eq.supply = eec - (a1 + a2 * eegp + a3 * cciutc);

   /* -------- Demand equation ------------- */
   eq.demand = eec - (b1 + b2 * eegp );

   /* -------- Instrumental variables -------*/
   lageegp = lag(eegp); lag2eegp=lag2(eegp);

   /* -------- Estimate parameters --------- */
   instruments _EXOG_ lageegp lag2eegp;
   fit supply demand / n3sls ;
   solve eegp eec / out=equilib;
run;
```

As a second example, suppose you want to compute points of intersection between the square root function and hyperbolas of the form $a + b/x$. That is, solve the system:

$$\text{(square root)} \qquad y = \sqrt{x}$$

$$\text{(hyperbola)} \qquad y = a + \frac{b}{x}$$

The following statements read parameters for several hyperbolas in the input data set TEST and solve the nonlinear equations. The SOLVEPRINT option on the SOLVE statement prints the solution values. The ID statement is used to include the values of A and B in the output of the SOLVEPRINT option.

```
data test;
  input a b @@;
  datalines;
  0 1   1 1   1 2
;

proc model data=test;
   eq.sqrt      = sqrt(x) - y;
   eq.hyperbola = a + b / x - y;
   solve x y / solveprint;
   id a b;
run;
```

The printed output produced by this example consists of a model summary report, a listing of the solution values for each observation, and a solution summary report. The model summary for this example is shown in Figure 20.9.

```
             Supply-Demand Model using General-form Equations

                           The MODEL Procedure

                              Model Summary

                        Model Variables         2
                        ID Variables            2
                        Equations               2
                        Number of Statements    2


                     Model Variables  x y
                           Equations  sqrt hyperbola
```

Figure 20.9. Model Summary Report

The output produced by the SOLVEPRINT option is shown in Figure 20.10.

```
                              The MODEL Procedure
                            Simultaneous Simulation

Observation  1  a                 0  b      1.0000  eq.hyperbola   0.000000
                Iterations       17  CC   0.000000


                                Solution Values

                                   x              y

                            1.000000       1.000000


Observation  2  a            1.0000  b      1.0000  eq.hyperbola   0.000000
                Iterations        5  CC   0.000000


                                Solution Values

                                   x              y

                            2.147899       1.465571


Observation  3  a            1.0000  b      2.0000  eq.hyperbola   0.000000
                Iterations        4  CC   0.000000


                                Solution Values

                                   x              y

                            2.875130       1.695621
```

Figure 20.10. Solution Values for Each Observation

For each observation, a heading line is printed that lists the values of the ID variables for the observation and information on the iterative process used to compute the solution. Following the heading line for the observation, the solution values are printed.

The heading line shows the solution method used (Newton's method by default), the number of iterations required, and the convergence measure, labeled CC=. This convergence measure indicates the maximum error by which solution values fail to satisfy the equations. When this error is small enough (as determined by the CONVERGE= option), the iterations terminate. The equation with the largest error is indicated in parentheses. For example, for observation 3 the HYPERBOLA equation has an error of 4.42×10^{-13} while the error of the SQRT equation is even smaller.

The last part of the SOLVE statement output is the solution summary report shown in Figure 20.11. This report summarizes the iteration history and the model solved.

```
                         The MODEL Procedure
                        Simultaneous Simulation

                           Data Set Options

                            DATA=    TEST


                           Solution Summary

                    Variables Solved            2
                    Implicit Equations          2
                    Solution Method        NEWTON
                    CONVERGE=                1E-8
                    Maximum CC            9.176E-9
                    Maximum Iterations         17
                    Total Iterations           26
                    Average Iterations    8.666667


                        Observations Processed

                             Read      3
                             Solved    3


               Variables Solved For    x y
               Equations Solved        sqrt hyperbola
```

Figure 20.11. Solution Summary Report

Monte Carlo Simulation

The RANDOM= option is used to request Monte Carlo (or stochastic) simulation to generate confidence intervals for a forecast. The confidence intervals are implied by the model's relationship to the the implicit random error term ϵ and the parameters.

The Monte Carlo simulation generates a random set of additive error values, one for each observation and each equation, and computes one set of perturbations of the parameters. These new parameters, along with the additive error terms, are then used to compute a new forecast that satisfies this new simultaneous system. Then a new set of additive error values and parameter perturbations is computed, and the process is repeated the requested number of times.

Consider the following exchange rate model for the U.S. dollar with the German mark and the Japanese yen:

$$rate\_jp = a_1 + b_1 im\_jp + c_1 di\_jp;$$

$$rate\_wg = a_2 + b_2 im\_wg + c_1 di\_wg;$$

where *rate_jp* and *rate_wg* are the exchange rate of the Japanese yen and the German mark versus the U.S. dollar respectively; *im_jp* and *im_wg* are the imports from Japan and Germany in 1984 dollars respectively; and *di_jp* and *di_wg* are the differences in inflation rate of Japan and the U.S., and Germany and the U.S. respectively. The

Monte Carlo capabilities of the MODEL procedure are used to generate error bounds on a forecast using this model.

```
proc model data=exchange;
   endo im_jp im_wg;
   exo di_jp di_wg;
   parms a1 a2 b1 b2 c1 c2;
   label rate_jp = 'Exchange Rate of Yen/$'
         rate_wg = 'Exchange Rate of Gm/$'
         im_jp = 'Imports to US from Japan in 1984 $'
         im_wg = 'Imports to US from WG in 1984 $'
         di_jp = 'Difference in Inflation Rates US-JP'
         di_wg = 'Difference in Inflation Rates US-WG';

   rate_jp = a1 + b1*im_jp + c1*di_jp;
   rate_wg = a2 + b2*im_wg + c2*di_wg;

            /* Fit the EXCHANGE data */
   fit rate_jp rate_wg / sur outest=xch_est outcov outs=s;

            /* Solve using the WHATIF data set */
   solve rate_jp rate_wg / data=whatif estdata=xch_est sdata=s
          random=100 seed=123 out=monte forecast;
   id yr;
   range yr=1986;
run;
```

Data for the EXCHANGE data set was obtained from the Department of Commerce and the yearly "Economic Report of the President."

First, the parameters are estimated using SUR selected by the SUR option on the FIT statement. The OUTEST= option is used to create the XCH_EST data set which contains the estimates of the parameters. The OUTCOV option adds the covariance matrix of the parameters to the XCH_EST data set. The OUTS= option is used to save the covariance of the equation error in the data set S.

Next, Monte Carlo simulation is requested using the RANDOM= option on the SOLVE statement. The data set WHATIF, shown below, is used to drive the forecasts. The ESTDATA= option reads in the XCH_EST data set which contains the parameter estimates and covariance matrix. Because the parameter covariance matrix is included, perturbations of the parameters are performed. The SDATA= option causes the Monte Carlo simulation to use the equation error covariance in the S data set to perturb the equation errors. The SEED= option selects the number 123 as seed value for the random number generator. The output of the Monte Carlo simulation is written to the data set MONTE selected by the OUT= option.

```
/* data for simulation */
data whatif;
   input yr rate_jp rate_wg imn_jp imn_wg emp_us emp_jp
      emp_wg  prod_us prod_jp prod_wg cpi_us cpi_jp cpi_wg;
   label cpi_us = 'US CPI 1982-1984 = 100'
```

```
              cpi_jp = ’JP CPI 1982-1984 = 100’
              cpi_wg = ’WG CPI 1982-1984 = 100’;
        im_jp = imn_jp/cpi_us;
        im_wg = imn_wg/cpi_us;
        ius = 100*(cpi_us-(lag(cpi_us)))/(lag(cpi_us));
        ijp = 100*(cpi_jp-(lag(cpi_jp)))/(lag(cpi_jp));
        iwg = 100*(cpi_wg-(lag(cpi_wg)))/(lag(cpi_wg));
        di_jp = ius - ijp;
        di_wg = ius - iwg;
     datalines;
     1980 226.63 1.8175 30714 11693 103.3 101.3 100.4 101.7
          125.4 109.8  .824  .909  .868
     1981 220.63 2.2631 35000 11000 102.8 102.2  97.9 104.6
          126.3 112.8  .909  .954  .922
     1982 249.06 2.4280 40000 12000  95.8 101.4  95.0 107.1
          146.8 113.3  .965  .980  .970
     1983 237.55 2.5539 45000 13100  94.4 103.4  91.1 111.6
          152.8 116.8  .996  .999 1.003
     1984 237.45 2.8454 50000 14300  99.0 105.8  90.4 118.5
          152.2 124.7 1.039 1.021 1.027
     1985 238.47 2.9419 55000 15600  98.1 107.6  91.3 124.2
          161.1 128.5 1.076 1.042 1.048
     1986    .      .   60000 17000  96.8 107.3  92.7 128.8
          163.8 130.7 1.096 1.049 1.047
     1987    .      .   65000 18500  97.1 106.1  92.8 132.0
          176.5 129.9 1.136 1.050 1.049
     1988    .      .   70000 20000  99.6 108.8  92.7 136.2
          190.0 135.9 1.183 1.057 1.063
     ;
```

To generate a confidence interval plot for the forecast, use PROC UNIVARIATE to generate percentile bounds and use PROC GPLOT to plot the graph. The following SAS statements produce the graph in Figure 20.12.

```
     proc sort data=monte;
        by yr;
     run;

     proc univariate data=monte noprint;
        by yr;
        var rate_jp rate_wg;
        output out=bounds mean=mean p5=p5 p95=p95;
     run;

     title "Monte Carlo Generated Confidence
                Intervals on a Forecast";
     proc gplot data=bounds;
        plot mean*yr p5*yr p95*yr /overlay;
        symbol1 i=join value=triangle;
        symbol2 i=join value=square l=4;
        symbol3 i=join value=square l=4;
     run;
```

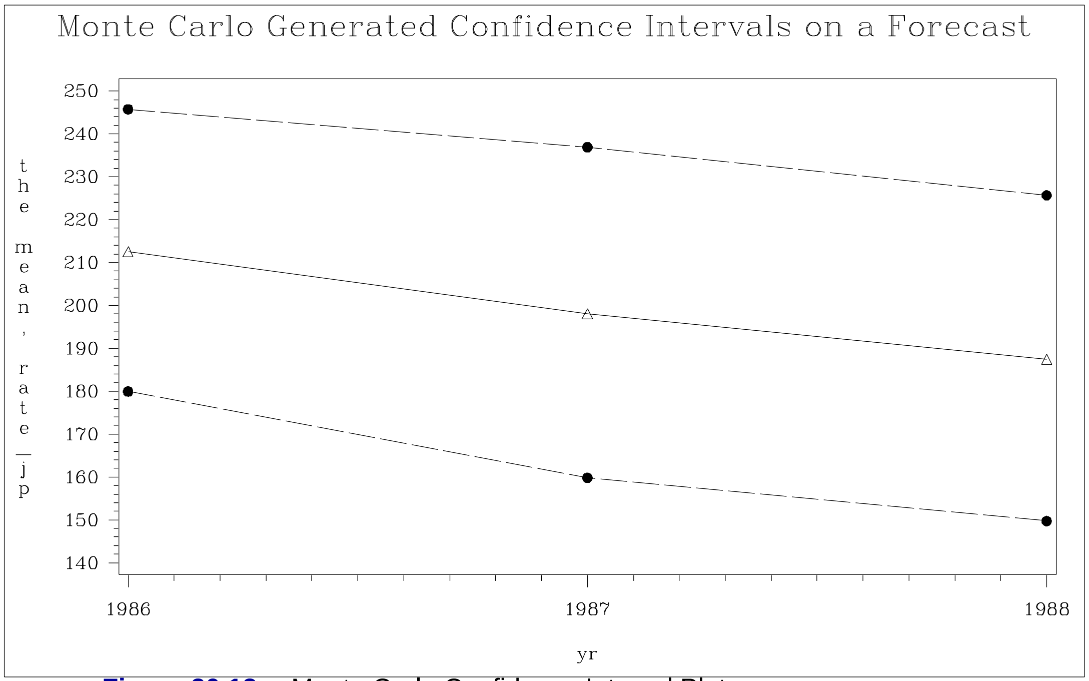


Figure 20.12. Monte Carlo Confidence Interval Plot

Syntax

The following statements can be used with the MODEL procedure:

PROC MODEL *options*;
  ABORT ;
  ARRAY *arrayname variables . . .* **;**
  ATTRIB *variable-list attribute-list [variable-list attribute-list]*;
  BOUNDS *bound1, bound2 . . .* **;**
  BY *variables*;
  CALL *name [(expression [, expression . . .])]* **;**
  CONTROL *variable [value] . . .* **;**
  DELETE ;
  DO *[variable = expression [TO expression] [BY expression]*
    [, expression **TO** *expression] [* **BY** *expression] . . .]*
   [**WHILE** *expression] [* **UNTIL** *expression]* **;**
  END ;
  DROP *variable . . .* **;**
  ENDOGENOUS *variable [initial values] . . .* **;**
  ERRORMODEL *equation-name ~ distribution*
    [**CDF=(** *CDF(options))]* **;**
  ESTIMATE *item [, item . . .] [,/ options]* **;**
  EXOGENOUS *variable [initial values] . . .* **;**
  FIT *equations [* **PARMS**=*(parameter values . . .)]*
    START=*(parameter values . . .)*
    [**DROP**=*(parameters)] [/ options]*;

FORMAT *variables [format] [* **DEFAULT** *= default-format]***;**
GOTO *statement_label* **;**
ID *variables***;**
IF *expression* **;**
IF *expression* **THEN** *programming_statement* **;**
  ELSE *programming_statement* **;**
variable **=** *expression* **;**
variable **+** *expression* **;**
INCLUDE *model files . . .* **;**
INSTRUMENTS *[instruments] [***_EXOG_** *]*
    *[***EXCLUDE***=(parameters)] [/ options]* **;**
KEEP *variable . . .* **;**
LABEL *variable ='label' . . .* **;**
LENGTH *variables [***$** *] length . . . [***DEFAULT***=length]***;**
LINK *statement_label* **;**
OUTVARS *variable . . .* **;**
PARAMETERS *variable [value] variable [value] . . .* **;**
PUT *print_item . . . [@] [@@]* **;**
RANGE *variable [= first] [***TO** *last]***;**
RENAME *old-name =new-name . . . [old-name=new-name]***;**
RESET *options***;**
RESTRICT *restriction1 [, restriction2 . . .]* **;**
RETAIN *variables values [variables values. . .]* **;**
RETURN ;
SOLVE *variables [***SATISFY=***(equations)] [/ options]* **;**
SUBSTR*(variable, index, length) = expression* **;**
SELECT *[(expression)]* **;**
  OTHERWISE *programming_statement* **;**
STOP ;
TEST *[* **"***name***"** *] test1 [, test2 . . .] [***,/** *options]* **;**
VAR *variable [initial values] . . .* **;**
WEIGHT *variable***;**
WHEN *(expression) programming_statement* **;**

Functional Summary

The statements and options in the MODEL procedure are summarized in the following table.

Description	**Statement**	**Option**
Data Set Options		
specify the input data set for the variables	FIT, SOLVE	DATA=
specify the input data set for parameters	FIT, SOLVE	ESTDATA=
specify the method for handling missing values	FIT	MISSING=

Description	**Statement**	**Option**
specify the input data set for parameters	MODEL	PARMSDATA=
specify the output data set for residual, predicted, or actual values	FIT	OUT=
specify the output data set for solution mode results	SOLVE	OUT=
write the actual values to OUT= data set	FIT	OUTACTUAL
select all output options	FIT	OUTALL
write the covariance matrix of the estimates	FIT	OUTCOV
write the parameter estimates to a data set	FIT	OUTEST=
write the parameter estimates to a data set	MODEL	OUTPARMS=
write the observations used to start the lags	SOLVE	OUTLAGS
write the predicted values to the OUT= data set	FIT	OUTPREDICT
write the residual values to the OUT= data set	FIT	OUTRESID
write the covariance matrix of the equation errors to a data set	FIT	OUTS=
write the **S** matrix used in the objective function definition to a data set	FIT	OUTSUSED=
write the estimate of the variance matrix of the moment generating function	FIT	OUTV=
read the covariance matrix of the equation errors	FIT, SOLVE	SDATA=
read the covariance matrix for GMM and ITGMM	FIT	VDATA=
specify the name of the time variable	FIT, SOLVE, MODEL	TIME=
select the estimation type to read	FIT, SOLVE	TYPE=
General ESTIMATE Statement Options		
specify the name of the data set in which the estimate of the functions of the parameters are to be written	ESTIMATE	OUTEST=
write the covariance matrix of the functions of the parameters to the OUTEST= data set	ESTIMATE	OUTCOV
print the covariance matrix of the functions of the parameters	ESTIMATE	COVB
print the correlation matrix of the functions of the parameters	ESTIMATE	CORRB
Printing Options for FIT Tasks		
print the modified Breusch-Pagan test for heteroscedasticity	FIT	BREUSCH
print the Chow test for structural breaks	FIT	CHOW=
print collinearity diagnostics	FIT	COLLIN

Description	Statement	Option
print the correlation matrices	FIT	CORR
print the correlation matrix of the parameters	FIT	CORRB
print the correlation matrix of the residuals	FIT	CORRS
print the covariance matrices	FIT	COV
print the covariance matrix of the parameters	FIT	COVB
print the covariance matrix of the residuals	FIT	COVS
print Durbin-Watson *d* statistics	FIT	DW
print first-stage R^2 statistics	FIT	FSRSQ
print Godfrey's tests for autocorrelated residuals for each equation	FIT	GODFREY
print Hausman's specification test	FIT	HAUSMAN
print tests of normality of the model residuals	FIT	NORMAL
print the predictive Chow test for structural breaks	FIT	PCHOW=
specify all the printing options	FIT	PRINTALL
print White's test for heteroscedasticity	FIT	WHITE
Options to Control FIT Iteration Output		
print the inverse of the crossproducts Jacobian matrix	FIT	I
print a summary iteration listing	FIT	ITPRINT
print a detailed iteration listing	FIT	ITDETAILS
print the crossproduct Jacobian matrix	FIT	XPX
specify all the iteration printing-control options	FIT	ITALL
Options to Control the Minimization Process		
specify the convergence criteria	FIT	CONVERGE=
select the Hessian approximation used for FIML	FIT	HESSIAN=
specifies the local truncation error bound for the integration	FIT, SOLVE, MODEL	LTEBOUND=
specify the maximum number of iterations allowed	FIT	MAXITER=
specify the maximum number of subiterations allowed	FIT	MAXSUBITER=
select the iterative minimization method to use	FIT	METHOD=
specifies the smallest allowed time step to be used in the integration	FIT, SOLVE, MODEL	MINTIMESTEP=
modify the iterations for estimation methods that iterate the **S** matrix or the **V** matrix	FIT	NESTIT
specify the smallest pivot value	MODEL, FIT, SOLVE	SINGULAR

Description	Statement	Option
specify the number of minimization iterations to perform at each grid point	FIT	STARTITER=
specify a weight variable	WEIGHT	
Options to Read and Write Model Files		
read a model from one or more input model files	INCLUDE	MODEL=
suppress the default output of the model file	MODEL, RESET	NOSTORE
specify the name of an output model file	MODEL, RESET	OUTMODEL=
delete the current model	RESET	PURGE
Options to List or Analyze the Structure of the Model		
print a dependency structure of a model	MODEL	BLOCK
print a graph of the dependency structure of a model	MODEL	GRAPH
print the model program and variable lists	MODEL	LIST
print the derivative tables and compiled model program code	MODEL	LISTCODE
print a dependency list	MODEL	LISTDEP
print a table of derivatives	MODEL	LISTDER
print a cross-reference of the variables	MODEL	XREF
General Printing Control Options		
expand parts of the printed output	FIT, SOLVE	DETAILS
print a message for each statement as it is executed	FIT, SOLVE	FLOW
select the maximum number of execution errors that can be printed	FIT, SOLVE	MAXERRORS=
select the number of decimal places shown in the printed output	FIT, SOLVE	NDEC=
suppress the normal printed output	FIT, SOLVE	NOPRINT
specify all the noniteration printing options	FIT, SOLVE	PRINTALL
print the result of each operation as it is executed	FIT, SOLVE	TRACE
request a comprehensive memory usage summary	FIT, SOLVE, MODEL, RESET	MEMORYUSE
turns off the NOPRINT option	RESET	PRINT
Statements that Declare Variables		
associate a name with a list of variables and constants	ARRAY	
declare a variable to have a fixed value	CONTROL	

Description	Statement	Option
declare a variable to be a dependent or endogenous variable	ENDOGENOUS	
declare a variable to be an independent or exogenous variable	EXOGENOUS	
specify identifying variables	ID	
assign a label to a variable	LABEL	
select additional variables to be output	OUTVARS	
declare a variable to be a parameter	PARAMETERS	
force a variable to hold its value from a previous observation	RETAIN	
declare a model variable	VAR	
declare an instrumental variable	INSTRUMENTS	
omit the default intercept term in the instruments list	INSTRUMENTS	NOINT
General FIT Statement Options		
omit parameters from the estimation	FIT	DROP=
associate a *variable* with an initial value as a parameter or a constant	FIT	INITIAL=
bypass OLS to get initial parameter estimates for GMM, ITGMM, or FIML	FIT	NOOLS
bypass 2SLS to get initial parameter estimates for GMM, ITGMM, or FIML	FIT	NO2SLS
specify the parameters to estimate	FIT	PARMS=
request confidence intervals on estimated parameters	FIT	PRL=
select a grid search	FIT	START=
Options to Control the Estimation Method Used		
specify nonlinear ordinary least squares	FIT	OLS
specify iterated nonlinear ordinary least squares	FIT	ITOLS
specify seemingly unrelated regression	FIT	SUR
specify iterated seemingly unrelated regression	FIT	ITSUR
specify two-stage least squares	FIT	2SLS
specify iterated two-stage least squares	FIT	IT2SLS
specify three-stage least squares	FIT	3SLS
specify iterated three-stage least squares	FIT	IT3SLS
specify full information maximum likelihood	FIT	FIML
specify simulated method of moments	FIT	NDRAW
specify number of draws for the V matrix	FIT	NDRAWV

Description	Statement	Option
specify number of initial observations for SMM	FIT	NPREOBS
select the variance-covariance estimator used for FIML	FIT	COVBEST=
specify generalized method of moments	FIT	GMM
specify the kernel for GMM and ITGMM	FIT	KERNEL=
specify iterated generalized method of moments	FIT	ITGMM
specify the type of generalized inverse used for the covariance matrix	FIT	GINV=
specify the denominator for computing variances and covariances	FIT	VARDEF=
specify adding the variance adjustment for SMM	FIT	ADJSMMV
specify variance correction for heteroscedasticity	FIT	HCCME=
specify GMM variance under arbitrary weighting matrix	FIT	GENGMMV
specify GMM variance under optimal weighting matrix	FIT	NOGENGMMV
Solution Mode Options		
select a subset of the model equations	SOLVE	SATISFY=
solve only for missing variables	SOLVE	FORECAST
solve for all solution variables	SOLVE	SIMULATE
Solution Mode Options: Lag Processing		
use solved values in the lag functions	SOLVE	DYNAMIC
use actual values in the lag functions	SOLVE	STATIC
produce successive forecasts to a fixed forecast horizon	SOLVE	NAHEAD=
select the observation to start dynamic solutions	SOLVE	START=
Solution Mode Options: Numerical Methods		
specify the maximum number of iterations allowed	SOLVE	MAXITER=
specify the maximum number of subiterations allowed	SOLVE	MAXSUBITER=
specify the convergence criteria	SOLVE	CONVERGE=
compute a simultaneous solution using a Jacobi-like iteration	SOLVE	JACOBI

Description	Statement	Option
compute a simultaneous solution using a Gauss-Seidel-like iteration	SOLVE	SEIDEL
compute a simultaneous solution using Newton's method	SOLVE	NEWTON
compute a nonsimultaneous solution	SOLVE	SINGLE
Monte Carlo Simulation Options		
specify quasi-random number generator	SOLVE	QUASI=
specify pseudo-random number generator	SOLVE	PSUEDO=
repeat the solution multiple times	SOLVE	RANDOM=
initialize the pseudo-random number generator	SOLVE	SEED=
Solution Mode Printing Options		
print between data points integration values for the DERT. variables and the auxiliary variables	FIT, SOLVE, MODEL	INTGPRINT
print the solution approximation and equation errors	SOLVE	ITPRINT
print the solution values and residuals at each observation	SOLVE	SOLVEPRINT
print various summary statistics	SOLVE	STATS
print tables of Theil inequality coefficients	SOLVE	THEIL
specify all printing control options	SOLVE	PRINTALL
General TEST Statement Options		
specify that a Wald test be computed	TEST	WALD
specify that a Lagrange multiplier test be computed	TEST	LM
specify that a likelihood ratio test be computed	TEST	LR
requests all three types of tests	TEST	ALL
specify the name of an output SAS data set that contains the test results	TEST	OUT=
Miscellaneous Statements		
specify the range of observations to be used	RANGE	
subset the data set with *by* variables	BY	

PROC MODEL Statement

PROC MODEL *options;*

The following options can be specified in the PROC MODEL statement. All of the nonassignment options (the options that do not accept a value after an equal sign) can have NO prefixed to the option name in the RESET statement to turn the option off. The default case is not explicitly indicated in the discussion that follows. Thus, for example, the option DETAILS is documented in the following, but NODETAILS is not documented since it is the default. Also, the NOSTORE option is documented because STORE is the default.

Data Set Options

DATA= *SAS-data-set*

names the input data set. Variables in the model program are looked up in the DATA= data set and, if found, their attributes (type, length, label, format) are set to be the same as those in the input data set (if not previously defined otherwise). The values for the variables in the program are read from the input data set when the model is estimated or simulated by FIT and SOLVE statements.

OUTPARMS= *SAS-data-set*
writes the parameter estimates to a SAS data set. See "Output Data Sets" for details.

PARMSDATA= *SAS-data-set*
names the SAS data set that contains the parameter estimates. See "Input Data Sets" for details.

Options to Read and Write Model Files

MODEL= *model-name*
MODEL= *(model-list)*
reads the model from one or more input model files created by previous PROC MODEL executions. Model files are written by the OUTMODEL= option.

NOSTORE
suppresses the default output of the model file. This option is only applicable when FIT or SOLVE statements are not used, the MODEL= option is not used, and when a model is specified.

OUTMODEL= *model-name*
specifies the name of an output model file to which the model is to be written. Model files are stored as members of a SAS catalog, with the type MODEL.

V5MODEL= *model-name*
reads model files written by Version 5 of SAS/ETS software.

Options to List or Analyze the Structure of the Model

These options produce reports on the structure of the model or list the programming statements defining the models. These options are automatically reset (turned off) after the reports are printed. To turn these options back on after a RUN statement has been entered, use the RESET statement or specify the options on a FIT or SOLVE statement.

BLOCK
prints an analysis of the structure of the model given by the assignments to model variables appearing in the model program. This analysis includes a classification of model variables into endogenous (dependent) and exogenous (independent) groups based on the presence of the variable on the left-hand side of an assignment statement. The endogenous variables are grouped into simultaneously determined blocks. The dependency structure of the simultaneous blocks and exogenous variables is also printed. The BLOCK option cannot analyze dependencies implied by general form equations.

GRAPH
prints the graph of the dependency structure of the model. The GRAPH option also invokes the BLOCK option and produces a graphical display of the information listed by the BLOCK option.

LIST
prints the model program and variable lists, including the statements added by PROC MODEL and macros.

LISTALL
selects the LIST, LISTDEP, LISTDER, and LISTCODE options.

LISTCODE
prints the derivative tables and compiled model program code. LISTCODE is a debugging feature and is not normally needed.

LISTDEP
prints a report that lists for each variable in the model program the variables that depend on it and that it depends on. These lists are given separately for current-period values and for lagged values of the variables.

The information displayed is the same as that used to construct the BLOCK report but differs in that the information is listed for all variables (including parameters, control variables, and program variables), not just the model variables. Classification into endogenous and exogenous groups and analysis of simultaneous structure is not done by the LISTDEP report.

LISTDER
prints a table of derivatives for FIT and SOLVE tasks. (The LISTDER option is only applicable for the default NEWTON method for SOLVE tasks.) The derivatives table shows each nonzero derivative computed for the problem. The derivative listed can be a constant, a variable in the model program, or a special derivative variable created to hold the result of the derivative expression. This option is turned on by the LISTCODE and PRINTALL options.

XREF
prints a cross-reference of the variables in the model program showing where each variable was referenced or given a value. The XREF option is normally used in conjunction with the LIST option. A more detailed description is given in the "Diagnostics and Debugging" section.

General Printing Control Options

DETAILS
specifies the detailed printout. Parts of the printed output are expanded when the DETAILS option is specified. If ODS GRAPHICS=ON is selected the following additional graphs of the residuals are produced: ACF, PACF, IACF, white noise, and QQ plot versus the normal.

FLOW
prints a message for each statement in the model program as it is executed. This debugging option is needed very rarely and produces voluminous output.

MAXERRORS= *n*
specifies the maximum number of execution errors that can be printed. The default is MAXERRORS=50.

NDEC= *n*
specifies the precision of the format that PROC MODEL uses when printing various numbers. The default is NDEC=3, which means that PROC MODEL attempts to print values using the D format but ensures that at least three significant digits are shown. If the NDEC= value is greater than nine, the BEST. format is used. The smallest value allowed is NDEC=2.

The NDEC= option affects the format of most, but not all, of the floating point numbers that PROC MODEL can print. For some values (such as parameter estimates), a precision limit one or two digits greater than the NDEC= value is used. This option does not apply to the precision of the variables in the output data set.

NOPRINT
suppresses the normal printed output but does not suppress error listings. Using any other print option turns the NOPRINT option off. The PRINT option can be used with the RESET statement to turn off NOPRINT.

PRINTALL
turns on all the printing-control options. The options set by PRINTALL are DETAILS; the model information options LIST, LISTDEP, LISTDER, XREF, BLOCK, and GRAPH; the FIT task printing options FSRSQ, COVB, CORRB, COVS, CORRS, DW, and COLLIN; and the SOLVE task printing options STATS, THEIL, SOLVEPRINT, and ITPRINT.

TRACE
prints the result of each operation in each statement in the model program as it is executed, in addition to the information printed by the FLOW option. This debugging option is needed very rarely and produces voluminous output.

MEMORYUSE
prints a report of the memory required for the various parts of the analysis.

FIT Task Options

The following options are used in the FIT statement (parameter estimation) and can also be used in the PROC MODEL statement: COLLIN, CONVERGE=, CORR, CORRB, CORRS, COVB, COVBEST=, COVS, DW, FIML, FSRSQ, GMM, HESSIAN=, I, INTGPRINT, ITALL, ITDETAILS, ITGMM, ITPRINT, ITOLS, ITSUR, IT2SLS, IT3SLS, KERNEL=, LTEBOUND=, MAXITER=, MAXSUBITER=, METHOD=, MINTIMESTEP=, NESTIT, N2SLS, N3SLS, OLS, OUTPREDICT, OUTRESID, OUTACTUAL, OUTLAGS, OUTERRORS, OUTALL, OUTCOV, SINGULAR=, STARTITER=, SUR, TIME=, VARDEF, and XPX. See "FIT Statement Syntax" later in this chapter for a description of these options.

When used in the PROC MODEL or RESET statement, these are default options for subsequent FIT statements. For example, the statement

```
   proc model n2sls ... ;
```

makes two-stage least squares the default parameter estimation method for FIT statements that do not specify an estimation method.

SOLVE Task Options

The following options used in the SOLVE statement can also be used in the PROC MODEL statement: CONVERGE=, DYNAMIC, FORECAST, INTGPRINT, ITPRINT, JACOBI, LTEBOUND=, MAXITER=, MAXSUBITER=, MINTIMESTEP=, NAHEAD=, NEWTON, OUTPREDICT, OUTRESID, OUTACTUAL, OUTLAGS, OUTERRORS, OUTALL, SEED=, SEIDEL, SIMULATE, SINGLE, SINGULAR=, SOLVEPRINT, START=, STATIC, STATS, THEIL, TIME=, and TYPE=. See "SOLVE Statement Syntax" later in this chapter for a description of these options.

When used in the PROC MODEL or RESET statement, these options provide default values for subsequent SOLVE statements.

BOUNDS Statement

BOUNDS *bound1 [, bound2 ...] ;*

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the associated FIT statement (that is, to either the preceding FIT statement or, in the absence of a preceding FIT statement, to the following FIT statement). You can specify any number of BOUNDS statements.

Each *bound* is composed of parameters and constants and inequality operators:

item operator item [operator item [operator item . . .]]

Each *item* is a constant, the name of an estimated parameter, or a list of parameter names. Each *operator* is '<', '>', '<=', or '>='.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See the "RESTRICT Statement" section on page 1048 for more information on the computational details of estimation with inequality restrictions.

Lagrange multipliers are reported for all the active boundary constraints. In the printed output and in the OUTEST= data set, the Lagrange multiplier estimates are identified with the names BOUND1, BOUND2, and so forth. The probability of the Lagrange multipliers are computed using a beta distribution (LaMotte 1994). To give the constraints more descriptive names, use the RESTRICT statement instead of the BOUNDS statement.

The following BOUNDS statement constrains the estimates of the parameters A and B and the ten parameters P1 through P10 to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < a b p1-p10 < 1;
```

The following is an example of the use of the BOUNDS statement:

```
title 'Holzman Function (1969), Himmelblau No. 21, N=3';
data zero;
   do i = 1 to 99;
      output;
   end;
run;

proc model data=zero ;
   parms x1= 100 x2= 12.5 x3=  3;
   bounds .1 <= x1 <= 100,
           0 <= x2 <=  25.6,
           0 <= x3 <=   5;

   t = 2 / 3;
   u = 25 + (-50 * log(0.01 * i )) ** t;
   v = (u - x2) ** x3;
   w = exp(-v / x1);
   eq.foo = -.01 * i + w;

   fit foo / method=marquardt;
run;
```

```
                Holzman Function (1969), Himmelblau No. 21, N=3

                              The MODEL Procedure

                       Nonlinear OLS Parameter Estimates

                                         Approx                    Approx
          Parameter        Estimate     Std Err    t Value     Pr > |t|

          x1               49.99999           0          .            .
          x2                     25           0          .            .
          x3                    1.5           0          .            .


               Number of Observations     Statistics for System

               Used                  99    Objective      5.455E-18
               Missing                0    Objective*N      5.4E-16
```

Figure 20.13. Output from Bounded Estimation

BY Statement

BY *variables;*

A BY statement is used with the FIT statement to obtain separate estimates for observations in groups defined by the BY variables. Note that if an output model file is written, using the OUTMODEL= option, the parameter values stored are those from the last BY group processed. To save parameter estimates for each BY group, use the OUTEST= option in the FIT statement.

A BY statement is used with the SOLVE statement to obtain solutions for observations in groups defined by the BY variables. If the BY variables are identical in the DATA= data set and the ESTDATA= data set, then the two data sets are synchronized and the simulations are performed using the data and parameters for each BY group. This holds for BY variables in the SDATA= data set as well. If, at some point, the BY variables don't match, BY processing is abandoned in either the ESTDATA= data set or the SDATA= data set, whichever has the missing BY value. If the DATA= data set does not contain BY variables and the ESTDATA= data set or the SDATA= data set does, then BY processing is performed for the ESTDATA= data set and the SDATA= data set by reusing the data in the DATA= data set for each BY group.

CONTROL Statement

CONTROL *variable [value] ... ;*

The CONTROL statement declares control variables and specifies their values. A control variable is like a parameter except that it has a fixed value and is not estimated from the data. You can use control variables for constants in model equations that you may want to change in different solution cases. You can use control variables to vary the program logic. Unlike the retained variables, these values are fixed across iterations.

ENDOGENOUS Statement

ENDOGENOUS *variable [initial-values] ... ;*

The ENDOGENOUS statement declares model variables and identifies them as endogenous. You can declare model variables with an ENDOGENOUS statement instead of with a VAR statement to help document the model or to indicate the default solution variables. The variables declared endogenous are solved when a SOLVE statement does not indicate which variables to solve. Valid abbreviations for the ENDOGENOUS statement are ENDOG and ENDO.

The DEPENDENT statement is equivalent to the ENDOGENOUS statement and is provided for the convenience of non-econometric practitioners.

The ENDOGENOUS statement optionally provides initial values for lagged dependent variables. See "Lag Logic" in the "Functions Across Time" section for more information.

ERRORMODEL Statement

ERRORMODEL *equation-name* $\sim$ *distribution* **[CDF=(** *CDF(options)* **)] ;**

The ERRORMODEL statement is the mechanism for specifying the distribution of the residuals. You must specify the dependent/endogenous variables or general form model name, a tilde ($\sim$), and then a distribution with its parameters. The following options are used in the ERRORMODEL statement:

Options to Specify the Distribution

CAUCHY(<*location, scale>* **)**
specifies the Cauchy distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

CHISQUARED (*df < , nc>* **)**
specifies the χ^2 distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

GENERAL(*Likelihood, <parm1, parm2, . . . parmn***>** **)**
specifies minus a general log likelihood function that you construct using SAS programming statements. $parm1, parm2, ...parmn$ are optional parameters for this distribution and are used for documentation purposes only.

F(*ndf, ddf < , nc>* **)**
specifies the F distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

NORMAL($v_1\ v_2\ \ldots\ v_n$ **)**
: specifies a multivariate normal (Gaussian) distribution with mean 0 and variances v_1 through v_n.

POISSON(*mean* **)**
: specifies the Poisson distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

T($v_1 v_2 \cdots v_n$, *df* **)**
: specifies a multivariate t distribution with noncentrality 0, variance v_1 through v_n, and common degrees of freedom df.

UNIFORM(*<left, right>* **)**
: specifies the uniform distribution. This option is only supported for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

Options to Specify the CDF for Simulation

CDF=(*CDF(options)* **)**
: specifies the univariate distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another. The CDF can be any of the distributions from the previous section with the exception of the General Likelihood. In addition, you can specify the empirical distribution of the residuals.

EMPIRICAL= (<TAILS=*(options)***>)**
: uses the sorted residual data to create an empirical CDF.

TAILS=(*tail options* **)**
: specifies how to handle the tails in computing the inverse CDF from an empirical distribution, where *Tail options* are:

NORMAL	specifies the normal distribution to extrapolate the tails.
T(*df*)	specifies the t distribution to extrapolate the tails.
PERCENT= *p*	specifies the percent of the observations to use in constructing each tail. The default for the PERCENT= option is 10. A normal distribution or a t distribution is used to extrapolate the tails to infinity. The variance for the tail distributions is are obtained from the data so that the empirical CDF is continuous.

ESTIMATE Statement

ESTIMATE *item [, item ...] [,/ options] ;*

The ESTIMATE statement computes estimates of functions of the parameters.

The ESTIMATE statement refers to the parameters estimated by the associated FIT statement (that is, to either the preceding FIT statement or, in the absence of a preceding FIT statement, to the following FIT statement). You can use any number of ESTIMATE statements.

If you specify options on the ESTIMATE statement, a comma is required before the "/" character separating the test expressions from the options, since the "/" character can also be used within test expressions to indicate division. Each *item* is written as an optional name followed by an expression,

["name"] expression

where *"name"* is a string used to identify the estimate in the printed output and in the OUTEST= data set.

Expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as "=" or "<") and logical operators (such as “&”) cannot be used in ESTIMATE statement expressions. Parameters named in ESTIMATE expressions must be among the parameters estimated by the associated FIT statement.

You can use the following options in the ESTIMATE statement:

OUTEST=
specifies the name of the data set in which the estimate of the functions of the parameters are to be written. The format for this data set is identical to the OUTEST= data set for the FIT statement.

If you specify a *name* in the ESTIMATE statement, that name is used as the parameter name for the estimate in the OUTEST= data set. If no *name* is provided and the expression is just a symbol, the symbol name is used; otherwise, the string "_Estimate #" is used, where "#" is the variable number in the OUTEST= data set.

OUTCOV
writes the covariance matrix of the functions of the parameters to the OUTEST= data set in addition to the parameter estimates.

COVB
prints the covariance matrix of the functions of the parameters.

CORRB
prints the correlation matrix of the functions of the parameters.

The following is an example of the use of the ESTIMATE statement in a segmented model:

```
data a;
   input y x @@;
   datalines;
   .46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
   .78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
   .80 15 .78 16
   ;

title 'Segmented Model -- Quadratic with Plateau';
proc model data=a;
```

```
      x0 = -.5 * b / c;

      if x < x0 then y = a + b*x + c*x*x;
      else           y = a + b*x0 + c*x0*x0;

      fit y start=( a .45 b .5 c -.0025 );

      estimate 'Join point' x0 ,
               'plateau' a + b*x0 + c*x0**2 ;
   run;
```

```
                      Segmented Model -- Quadratic with Plateau

                                 The MODEL Procedure

                              Nonlinear OLS  Estimates

                                  Approx                  Approx
Term              Estimate       Std Err    t Value     Pr > |t|    Label

Join point         12.7504        1.2785       9.97       <.0001    x0
plateau           0.777516        0.0123      63.10       <.0001    a + b*x0 + c*x0**2
```

Figure 20.14. ESTIMATE Statement Output

EXOGENOUS Statement

EXOGENOUS *variable [initial-values] ... ;*

The EXOGENOUS statement declares model variables and identifies them as exogenous. You can declare model variables with an EXOGENOUS statement instead of with a VAR statement to help document the model or to indicate the default instrumental variables. The variables declared exogenous are used as instruments when an instrumental variables estimation method is requested (such as N2SLS or N3SLS) and an INSTRUMENTS statement is not used. Valid abbreviations for the EXOGENOUS statement are EXOG and EXO.

The INDEPENDENT statement is equivalent to the ENDOGENOUS statement and is provided for the convenience of non-econometric practitioners.

The EXOGENOUS statement optionally provides initial values for lagged exogenous variables. See "Lag Logic" in the "Functions Across Time" section for more information.

FIT Statement

FIT *[equations] [* **PARMS=(** *parameter [values] ...)]*

[**START=(** *parameter values ...)]*
[**DROP=(** *parameter ...)]*
[**INITIAL=(** *variable = [parameter* | *constant] ...)*
[/ options] **;**

The FIT statement estimates model parameters by fitting the model equations to input data and optionally selects the equations to be fit. If the list of equations is omitted, all model equations containing parameters are fit.

The following options can be used in the FIT statement.

DROP= *(parameters ...)*
specifies that the named parameters not be estimated. All the parameters in the equations fit are estimated except those listed in the DROP= option. The dropped parameters retain their previous values and are not changed by the estimation.

INITIAL= *(variable = [parameter | constant] ...)*
associates a *variable* with an initial value as a *parameter* or a *constant*.

PARMS= *(parameters [values] ...)*
selects a subset of the parameters for estimation. When the PARMS= option is used, only the named parameters are estimated. Any parameters not specified in the PARMS= list retain their previous values and are not changed by the estimation.

PRL= WALD | LR | BOTH
requests confidence intervals on estimated parameters. By default the PRL option produces 95% likelihood ratio confidence limits. The coverage of the confidence interval is controlled by the ALPHA= option in the FIT statement.

START= *(parameter values ...)*
supplies starting values for the parameter estimates. If the START= option specifies more than one starting value for one or more parameters, a grid search is performed over all combinations of the values, and the best combination is used to start the iterations. For more information, see the STARTITER= option.

Options to Control the Estimation Method Used

ADJSMMV
specifies adding the variance adjustment from simulating the moments to the variance-covariance matrix of the parameter estimators. By default no adjustment is made.

COVBEST=GLS | CROSS | FDA
specifies the variance-covariance estimator used for FIML. COVBEST=GLS selects the generalized least-squares estimator. COVBEST=CROSS selects the crossproducts estimator. COVBEST=FDA selects the inverse of the finite difference approximation to the Hessian. The default is COVBEST=CROSS.

FIML
specifies full information maximum likelihood estimation.

GINV=G2 | G4
specifies the type of generalized inverse to be used when computing the covariance matrix. G4 selects the Moore Penrose generalized inverse. The default is GINV=G2.

Rather than deleting linearly related rows and columns of the covariance matrix, the Moore-Penrose generalized inverse averages the variance effects between collinear rows. When the option GINV=G4 is used, the Moore-Penrose generalized inverse is use to calculate standard errors and the covariance matrix of the parameters as well as the change vector for the optimization problem. For singular systems, a normal G2 inverse is used to determine the singular rows so that the parameters can be marked in the Parameter Estimates table. Whether or not you use a G4 inverse, if the covariance matrix is singular, the parameter estimates are not unique. Refer to Noble and Daniel (1977, pp. 337–340) for more details on generalized inverses.

GENGMMV

specify GMM variance under arbitrary weighting matrix. See the "Estimation Methods" section for more details.

The is the default method for GMM estimation.

GMM

specifies generalized method of moments estimation.

HCCME= 0 | 1 | 2 | 3 | NO

specifies the type of heteroscedasticity-consistent covariance matrix estimator to use for OLS, 2SLS, 3SLS, SUR, and the iterated versions of these estimation methods. The number corresponds to the type of covariance matrix estimator to use as

$$\begin{array}{ll} HC_0: & \hat{\epsilon}_t^2 \\ HC_1: & \frac{n}{n-df}\hat{\epsilon}_t^2 \\ HC_2: & \hat{\epsilon}_t^2/(1-\hat{h}_t) \\ HC_3: & \hat{\epsilon}_t^2/(1-\hat{h}_t)^2 \end{array}$$

The default is NO.

ITGMM

specifies iterated generalized method of moments estimation.

ITOLS

specifies iterated ordinary least-squares estimation. This is the same as OLS unless there are cross-equation parameter restrictions.

ITSUR

specifies iterated seemingly unrelated regression estimation

IT2SLS

specifies iterated two-stage least-squares estimation. This is the same as 2SLS unless there are cross-equation parameter restrictions.

IT3SLS

specifies iterated three-stage least-squares estimation.

KERNEL=(PARZEN | BART | QS, *[c]*, *[e]* **)**
KERNEL=PARZEN | BART | QS

specifies the kernel to be used for GMM and ITGMM. PARZEN selects the Parzen kernel, BART selects the Bartlett kernel, and QS selects the Quadratic Spectral kernel. $e \geq 0$ and $c \geq 0$ are used to compute the bandwidth parameter. The default is KERNEL=(PARZEN, 1, 0.2). See the "Estimation Methods" section for more details.

N2SLS | 2SLS
specifies nonlinear two-stage least-squares estimation. This is the default when an INSTRUMENTS statement is used.

N3SLS | 3SLS
specifies nonlinear three-stage least-squares estimation.

NDRAW <=*number of draws*>
requests the simulation method for estimation. H is the *number of draws*. If *number of draws* is not specified, the default H is set to 10.

NOOLS
NO2SLS
specifies bypassing OLS or 2SLS to get initial parameter estimates for GMM, ITGMM, or FIML. This is important for certain models that are poorly defined in OLS or 2SLS, or if good initial parameter values are already provided. Note that for GMM, the V matrix is created using the initial values specified and this may not be consistently estimated.

NO3SLS
specifies not to use 3SLS automatically for FIML initial parameter starting values.

NOGENGMMV
specify not to use GMM variance under arbitrary weighting matrix. Use GMM variance under optimal weighting matrix instead. See the "Estimation Methods" section for more details.

NPREOBS =*number of obs to initialize*
specifies the initial number of observations to run the simulation before the simulated values are compared to observed variables. This option is most useful in cases where the program statements involve lag operations. Use this option to avoid the effect of the starting point on the simulation.

NVDRAW =*number of draws for V matrix*
specifies H', the *number of draws for V matrix*. If this option is not specified, the default H' is set to 20.

OLS
specifies ordinary least-squares estimation. This is the default when no INSTRUMENTS statement is used.

SUR
specifies seemingly unrelated regression estimation.

VARDEF=N | WGT | DF | WDF
specifies the denominator to be used in computing variances and covariances. VARDEF=N specifies that the number of nonmissing observations be used. VARDEF=WGT specifies that the sum of the weights be used. VARDEF=DF specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used. VARDEF=WDF specifies that the sum of the weights minus the model degrees of freedom be used. The default is VARDEF=DF. VARDEF=N is used for FIML estimation.

Data Set Options

DATA= *SAS-data-set*

specifies the input data set. Values for the variables in the program are read from this data set. If the DATA= option is not specified on the FIT statement, the data set specified by the DATA= option on the PROC MODEL statement is used.

ESTDATA= *SAS-data-set*

specifies a data set whose first observation provides initial values for some or all of the parameters.

MISSING= PAIRWISE | DELETE

The option MISSING=PAIRWISE specifies that missing values are tracked on an equation-by-equation basis. The MISSING=DELETE option specifies that the entire observation is omitted from the analysis when any equation has a missing predicted or actual value for the equation. The default is MISSING=DELETE.

OUT= *SAS-data-set*

names the SAS data set to contain the residuals, predicted values, or actual values from each estimation. Only the residuals are output by default.

OUTACTUAL

writes the actual values of the endogenous variables of the estimation to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTALL

selects the OUTACTUAL, OUTERRORS, OUTLAGS, OUTPREDICT, and OUTRESID options.

OUTCOV
COVOUT

writes the covariance matrix of the estimates to the OUTEST= data set in addition to the parameter estimates. The OUTCOV option is applicable only if the OUTEST= option is also specified.

OUTEST= *SAS-data-set*

names the SAS data set to contain the parameter estimates and optionally the covariance of the estimates.

OUTLAGS

writes the observations used to start the lags to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTPREDICT

writes the predicted values to the OUT= data set. This option is applicable only if OUT= is specified.

OUTRESID

writes the residual values computed from the parameter estimates to the OUT= data set. The OUTRESID option is the default if neither OUTPREDICT nor OUTACTUAL is specified. This option is applicable only if the OUT= option is specified.

OUTS= *SAS-data-set*
names the SAS data set to contain the estimated covariance matrix of the equation errors. This is the covariance of the residuals computed from the parameter estimates.

OUTSN= *SAS-data-set*
names the SAS data set to contain the estimated normalized covariance matrix of the equation errors. This is valid for multivariate t-distribution estimation.

OUTSUSED= *SAS-data-set*
names the SAS data set to contain the S matrix used in the objective function definition. The OUTSUSED= data set is the same as the OUTS= data set for the methods that iterate the S matrix.

OUTUNWGTRESID
writes the unweighted residual values computed from the parameter estimates to the OUT= data set. These are residuals computed as $actual - predicted$ with no accounting for the WEIGHT statement, the _WEIGHT_ variable, or any variance expressions. This option is applicable only if the OUT= option is specified.

OUTV= *SAS-data-set*
names the SAS data set to contain the estimate of the variance matrix for GMM and ITGMM.

SDATA= *SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The matrix read from the SDATA= data set is used for the equation covariance matrix (S matrix) in the estimation. (The SDATA= S matrix is used to provide only the initial estimate of **S** for the methods that iterate the S matrix.)

TIME= *name*
specifies the name of the time variable. This variable must be in the data set.

TYPE= *name*
specifies the estimation type to read from the SDATA= and ESTDATA= data sets. The name specified in the TYPE= option is compared to the _TYPE_ variable in the ESTDATA= and SDATA= data sets to select observations to use in constructing the covariance matrices. When the TYPE= option is omitted, the last estimation type in the data set is used. Valid values are the estimation methods used in PROC MODEL.

VDATA= *SAS-data-set*
specifies a data set containing a variance matrix for GMM and ITGMM estimation.

Printing Options for FIT Tasks

BREUSCH= *(variable-list)*
specifies the modified Breusch-Pagan test, where *variable-list* is a list of variables used to model the error variance.

CHOW= *obs*
CHOW= *(obs1 obs2 ... obsn)*
prints the Chow test for break points or structural changes in a model. The argument is the number of observations in the first sample or a parenthesized list of first sample sizes. If the size of the one of the two groups in which the sample is partitioned is

less than the number of parameters, then a Predictive Chow test is automatically used. See the section "Chow Tests" on page 1131 for details.

COLLIN
prints collinearity diagnostics for the Jacobian crossproducts matrix (XPX) after the parameters have converged. Collinearity diagnostics are also automatically printed if the estimation fails to converge.

CORR
prints the correlation matrices of the residuals and parameters. Using CORR is the same as using both CORRB and CORRS.

CORRB
prints the correlation matrix of the parameter estimates.

CORRS
prints the correlation matrix of the residuals.

COV
prints the covariance matrices of the residuals and parameters. Specifying COV is the same as specifying both COVB and COVS.

COVB
prints the covariance matrix of the parameter estimates.

COVS
prints the covariance matrix of the residuals.

DW <=>
prints Durbin-Watson d statistics, which measure autocorrelation of the residuals. When the residual series is interrupted by missing observations, the Durbin-Watson statistic calculated is d' as suggested by Savin and White (1978). This is the usual Durbin-Watson computed by ignoring the gaps. Savin and White show that it has the same null distribution as the DW with no gaps in the series and can be used to test for autocorrelation using the standard tables. The Durbin-Watson statistic is not valid for models containing lagged endogenous variables.

You can use the DW= option to request higher order Durbin-Watson statistics. Since the ordinary Durbin-Watson statistic tests only for first-order autocorrelation, the Durbin-Watson statistics for higher-order autocorrelation are called *generalized Durbin-Watson* statistics.

DWPROB
Use the DWPROB option to print the significance level (p-values) for the Durbin-Watson tests. Since the Durbin-Watson p-values are computationally expensive, they are not reported by default. In the Durbin-Watson test, the null hypothesis is that there is autocorrelation at a specific lag.

See the section "Generalized Durbin-Watson Tests" in the Autoreg Chapter for limitations of the statistic.

FSRSQ
prints the first-stage R^2 statistics for instrumental estimation methods. These R^2s measure the proportion of the variance retained when the Jacobian columns associated with the parameters are projected through the instruments space.

GODFREY
GODFREY= *n*
performs Godfrey's tests for autocorrelated residuals for each equation, where *n* is the maximum autoregressive order, and specifies that Godfrey's tests be computed for lags 1 through *n*. The default number of lags is one.

HAUSMAN
performs Hausman's specification test, or m-statistics.

NORMAL
performs tests of normality of the model residuals.

PCHOW= *obs*
PCHOW= *(obs1 obs2 ... obsn)*
prints the Predictive Chow test for break points or structural changes in a model. The argument is the number of observations in the first sample or a parenthesized list of first sample sizes. See the section "Chow Tests" on page 1131 for details.

PRINTALL
specifies the printing options COLLIN, CORRB, CORRS, COVB, COVS, DETAILS, DW, and FSRSQ.

WHITE
specifies White's test.

Options to control iteration output

Details of the output produced are discussed in the section "Iteration History".

I
prints the inverse of the crossproducts Jacobian matrix at each iteration.

ITALL
specifies all iteration printing-control options (I, ITDETAILS, ITPRINT, and XPX). ITALL also prints the crossproducts matrix (labeled CROSS), the parameter change vector, and the estimate of the cross-equation covariance of residuals matrix at each iteration.

ITDETAILS
prints a detailed iteration listing. This includes the ITPRINT information and additional statistics.

ITPRINT
prints the parameter estimates, objective function value, and convergence criteria at each iteration.

XPX
prints the crossproducts Jacobian matrix at each iteration.

Options to Control the Minimization Process

The following options may be helpful when you experience a convergence problem:

CONVERGE= *value1*
CONVERGE= *(value1, value2)*
specifies the convergence criteria. The convergence measure must be less than *value1* before convergence is assumed. *value2* is the convergence criterion for the **S** and **V** matrices for **S** and **V** iterated methods. *value2* defaults to *value1*. See "The Convergence Criteria" for details. The default value is CONVERGE=.001.

HESSIAN= CROSS | GLS | FDA
specifies the Hessian approximation used for FIML. HESSIAN=CROSS selects the crossproducts approximation to the Hessian, HESSIAN=GLS selects the generalized least-squares approximation to the Hessian, and HESSIAN=FDA selects the finite difference approximation to the Hessian. HESSIAN=GLS is the default.

LTEBOUND= *n*
specifies the local truncation error bound for the integration. This option is ignored if no ODE's are specified.

MAXITER= *n*
specifies the maximum number of iterations allowed. The default is MAXITER=100.

MAXSUBITER= *n*
specifies the maximum number of subiterations allowed for an iteration. For the GAUSS method, the MAXSUBITER= option limits the number of step halvings. For the MARQUARDT method, the MAXSUBITER= option limits the number of times λ can be increased. The default is MAXSUBITER=30. See "Minimization Methods" for details.

METHOD= GAUSS | MARQUARDT
specifies the iterative minimization method to use. METHOD=GAUSS specifies the Gauss-Newton method, and METHOD=MARQUARDT specifies the Marquardt-Levenberg method. The default is METHOD=GAUSS. See "Minimization Methods" for details.

MINTIMESTEP= *n*
specifies the smallest allowed time step to be used in the integration. This option is ignored if no ODE's are specified.

NESTIT
changes the way the iterations are performed for estimation methods that iterate the estimate of the equation covariance (**S** matrix). The NESTIT option is relevant only for the methods that iterate the estimate of the covariance matrix (ITGMM, ITOLS, ITSUR, IT2SLS, IT3SLS). See "Details on the Covariance of Equation Errors" for an explanation of NESTIT.

SINGULAR= *value*
specifies the smallest pivot value allowed. The default 1.0E-12.

STARTITER= *n*
specifies the number of minimization iterations to perform at each grid point. The default is STARTITER=0, which implies that no minimization is performed at the grid points. See "Using the STARTITER option" for more details.

Other Options

Other options that can be used on the FIT statement include the following that list and analyze the model: BLOCK, GRAPH, LIST, LISTCODE, LISTDEP, LISTDER, and XREF. The following printing control options are also available: DETAILS, FLOW, INTGPRINT, MAXERRORS=, NOPRINT, PRINTALL, and TRACE. For complete descriptions of these options, see the discussion of the PROC MODEL statement options earlier in this chapter.

ID Statement

ID *variables;*

The ID statement specifies variables to identify observations in error messages or other listings and in the OUT= data set. The ID variables are normally SAS date or datetime variables. If more than one ID variable is used, the first variable is used to identify the observations; the remaining variables are added to the OUT= data set.

INCLUDE Statement

INCLUDE *model-names ... ;*

The INCLUDE statement reads model files and inserts their contents into the current model. However, instead of replacing the current model as the RESET MODEL= option does, the contents of included model files are inserted into the model program at the position that the INCLUDE statement appears.

INSTRUMENTS Statement

The INSTRUMENTS statement specifies the instrumental variables to be used in the N2SLS, N3SLS, IT2SLS, IT3SLS, GMM, and ITGMM estimation methods. There are three forms of the INSTRUMENTS statement:

INSTRUMENTS *variables [* **_EXOG_** *] ;*

INSTRUMENTS *[instruments] [* **_EXOG_** *]*
[**EXCLUDE**=*(parameters)] [/ options]* **;**

INSTRUMENTS *(equation, variables) (equation, variables) ... ;*

The first form of the INSTRUMENTS statement is used only before a FIT statement and defines the default instruments list. The items specified as instruments can be variables or the special keyword _EXOG_. _EXOG_ indicates that all the model variables declared EXOGENOUS are to be added to the instruments list.

The second form of the INSTRUMENTS statement is used only after the FIT statement and before the next RUN statement. The items specified as instruments for the second form can be variables, names of parameters to be estimated, or the special keyword _EXOG_. If you specify the name of a parameter in the instruments list, the partial derivatives of the equations with respect to the parameter (that is, the columns of the Jacobian matrix associated with the parameter) are used as instruments. The parameter itself is not used as an instrument. These partial derivatives should not depend on any of the parameters to be estimated. Only the names of parameters to be estimated can be specified.

EXCLUDE= _(parameters)_

specifies that the derivatives of the equations with respect to all of the parameters to be estimated, except the parameters listed in the EXCLUDE list, be used as instruments, in addition to the other instruments specified. If you use the EXCLUDE= option, you should be sure that the derivatives with respect to the nonexcluded parameters in the estimation are independent of the endogenous variables and not functions of the parameters estimated.

A third form of the INSTRUMENTS statement is used to specify instruments for each equation. There is no explicit intercept added, parameters cannot be specified to represent instruments, and the _EXOG_ keyword is not allowed. Equations not explicitly assigned instruments will use all the instruments specified for the other equations as well as instruments not assigned specific equations. In the following example, z1, z2, and z3 are instruments used with equation y1, and z2, z3, and z4 are instruments used with equation y2.

```
proc model data=data_sim;
   exogenous x1 x2;
   parms a b c d e f;

   y1 =a*x1**2 + b*x2**2 + c*x1*x2   ;
   y2 =d*x1**2 + e*x2**2 + f*x1*x2**2;

   fit y1 y2 / 3sls ;
   instruments (y1, z1 z2 z3) (y2,z2 z3 z4);
run;
```

The following option is specified on the INSTRUMENTS statement following a slash (/):

NOINTERCEPT
NOINT

excludes the constant of 1.0 (intercept) from the instruments list. An intercept is always included as an instrument unless NOINTERCEPT is specified.

When a FIT statement specifies an instrumental variables estimation method and no INSTRUMENTS statement accompanies the FIT statement, the default instruments are used. If no default instruments list has been specified, all the model variables declared EXOGENOUS are used as instruments. See the section "Choice of Instruments" on page 1135 for more details.

INTONLY

INTONLY

specifies that only the intercept be used as an instrument. This option is used for GMM estimation where the moments have been specified explicitly.

LABEL Statement

LABEL *variable='label' ... ;*

The LABEL statement specifies a label of up to 255 characters for parameters and other variables used in the model program. Labels are used to identify parts of the printout of FIT and SOLVE tasks. The labels will be displayed in the output if the LINESIZE= option is large enough.

MOMENT Statement

In many scenarios, endogenous variables are observed from data. From the models we can simulate these endogenous variables based on a fixed set of parameters. The goal of SMM is to find a set of parameters such that the moments of the simulated data match the moments of the observed variables. If there are many moments to match, the code may be tedious. The following MOMENT statement provides a way to generate some commonly used moments automatically. Multiple MOMENT statements can be used.

MOMENT *variables = moment specification ;*

variables can be one or more endogenous variables.

moment specification can have the following four types:

1. (*number list*) – the endogenous variable is raised to the power specified by each number in *number list.* For example,

```
moment y = (2 3);
```

adds the following two equations to be estimated:

```
eq._moment_1 = y**2 - pred.y**2;
eq._moment_2 = y**3 - pred.y**3;
```

2. ABS(*number list*) – the absolute value of the endogenous variable is raised to the power specified by each number in *number list.* For example,

```
moment y = ABS(3);
```

adds the following equation to be estimated:

```
eq._moment_2 = abs(y)**3 - abs(pred.y)**3;
```

3. LAG*num* (*number list*) – the endogenous variable is multiplied by the *num*th lag of the endogenous variable, and this product is raised to the power specified by each number in *number list.* For example,

```
moment y = LAG4(3);
```

adds the following equation to be estimated:

```
eq._moment_3 = (y*lag4(y))**3 - (pred.y*lag4(pred.y))**3;
```

4. ABS_LAG*num* (*number list*) – the endogenous variable is multiplied by the *num*th lag of the endogenous variable, and the absolute value of this product is raised to the power specified by each number in *number list.* For example,

```
moment y = ABS_LAG4(3);
```

adds the following equation to be estimated:

```
eq._moment_4 = abs(y*lag4(y))**3 - abs(pred.y*lag4(pred.y))**3;
```

The following PROC MODEL code utilizes the MOMENT statement to generate 24 moments and fit these moments using SMM.

```
 proc model data=_tmpdata list;
       parms a b .5 s 1;
       instrument _exog_ / intonly;

       u = rannor( 10091 );
       z = rannor( 97631 );

       lsigmasq = xlag(sigmasq,exp(a));

       lnsigmasq = a + b * log(lsigmasq) + s * u;
       sigmasq = exp( lnsigmasq );

       y = sqrt(sigmasq) * z;

       moment y = (2 4) abs(1 3) abs_lag1(1 2) abs_lag2(1 2);
       moment y = abs_lag3(1 2) abs_lag4(1 2)
                  abs_lag5(1 2) abs_lag6(1 2)
                  abs_lag7(1 2) abs_lag8(1 2)
                  abs_lag9(1 2) abs_lag10(1 2);

       fit y  / gmm npreobs=20 ndraw=10;
       bound s > 0, 1>b>0;

run;
```

OUTVARS Statement

OUTVARS *variables;*

The OUTVARS statement specifies additional variables defined in the model program to be output to the OUT= data sets. The OUTVARS statement is not needed unless the variables to be added to the output data set are not referred to by the model, or unless you wish to include parameters or other special variables in the OUT= data set. The OUTVARS statement includes additional variables, whereas the KEEP statement excludes variables.

PARAMETERS Statement

PARAMETERS *variable [value] [variable [value]] ... ;*

The PARAMETERS statement declares the parameters of a model and optionally sets their initial values. Valid abbreviations are PARMS and PARM.

Each parameter has a single value associated with it, which is the same for all observations. Lagging is not relevant for parameters. If a value is not specified in the PARMS statement (or by the PARMS= option of a FIT statement), the value defaults to 0.0001 for FIT tasks and to a missing value for SOLVE tasks.

RANGE Statement

RANGE *variable [= first] [***TO** *last];*

The RANGE statement specifies the range of observations to be read from the DATA= data set. For FIT tasks, the RANGE statement controls the period of fit for the estimation. For SOLVE tasks, the RANGE statement controls the simulation period or forecast horizon.

The RANGE variable must be a numeric variable in the DATA= data set that identifies the observations, and the data set must be sorted by the RANGE variable. The first observation in the range is identified by *first*, and the last observation is identified by *last*.

PROC MODEL uses the first *l* observations prior to *first* to initialize the lags, where *l* is the maximum number of lags needed to evaluate any of the equations to be fit or solved, or the maximum number of lags needed to compute any of the instruments when an instrumental variables estimation method is used. There should be at least *l* observations in the data set before *first*. If *last* is not specified, all the nonmissing observations starting with *first* are used.

If *first* is omitted, the first *l* observations are used to initialize the lags, and the rest of the data, until *last*, is used. If a RANGE statement is used but both *first* and *last* are omitted, the RANGE statement variable is used to report the range of observations processed.

The RANGE variable should be nonmissing for all observations. Observations containing missing RANGE values are deleted.

The following are examples of RANGE statements:

```
range year = 1971 to 1988;              /* yearly  data  */
range date = '1feb73'd to '1nov82'd;    /* monthly data  */
range time = 60.5;                      /* time in years */
range year to 1977;         /* use all years through 1977 */
range date; /* use values of date to report period-of-fit */
```

RESET Statement

RESET *options;*

All of the options of the PROC MODEL statement can be reset by the RESET statement. In addition, the RESET statement supports one additional option:

PURGE
deletes the current model so that a new model can be defined.

When the MODEL= option is used in the RESET statement, the current model is deleted before the new model is read.

RESTRICT Statement

RESTRICT *restriction1 [, restriction2 ...] ;*

The RESTRICT statement is used to impose linear and nonlinear restrictions on the parameter estimates.

RESTRICT statements refer to the parameters estimated by the associated FIT statement (that is, to either the preceding FIT statement or, in the absence of a preceding FIT statement, to the following FIT statement). You can specify any number of RESTRICT statements.

Each *restriction* is written as an optional name, followed by an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

["name"] expression operator expression

The optional *"name"* is a string used to identify the restriction in the printed output and in the OUTEST= data set. The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional, as in the TEST statement (=0).

Restriction expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as "=" or "<") and logical operators (such as “&”) cannot be used in RESTRICT statement expressions. Parameters named in restriction expressions must be among the parameters estimated by the associated FIT statement. Expressions can refer to variables defined in the program.

The restriction expressions can be linear or nonlinear functions of the parameters.

The following is an example of the use of the RESTRICT statement:

```
proc model data=one;
   endogenous y1 y2;
   exogenous x1 x2;
   parms a b c;
   restrict b*(b+c) <= a;

   eq.one = -y1/c + a/x2 + b * x1**2 + c * x2**2;
   eq.two = -y2 * y1 + b * x2**2 - c/(2 * x1);

   fit one two / fiml;
run;
```

SOLVE Statement

SOLVE *[variables]* [**SATISFY=** *equations]* [**INITIAL=** *(variable=[parameter]]* *[/options];*

The SOLVE statement specifies that the model be simulated or forecast for input data values and, optionally, selects the variables to be solved. If the list of variables is omitted, all of the model variables declared ENDOGENOUS are solved. If no model variables are declared ENDOGENOUS, then all model variables are solved.

The following specification can be used in the SOLVE statement:

SATISFY= *equation*
SATISFY= *(equations)*
specifies a subset of the model equations that the solution values are to satisfy. If the SATISFY= option is not used, the solution is computed to satisfy all the model equations. Note that the number of equations must equal the number of variables solved.

Data Set Options

DATA= *SAS-data-set*
names the input data set. The model is solved for each observation read from the DATA= data set. If the DATA= option is not specified on the SOLVE statement, the data set specified by the DATA= option on the PROC MODEL statement is used.

ESTDATA= *SAS-data-set*
names a data set whose first observation provides values for some or all of the parameters and whose additional observations (if any) give the covariance matrix of the parameter estimates. The covariance matrix read from the ESTDATA= data set is used to generate multivariate normal pseudo-random shocks to the model parameters when the RANDOM= option requests Monte Carlo simulation.

OUT= *SAS-data-set*
outputs the predicted (solution) values, residual values, actual values, or equation errors from the solution to a data set. Only the solution values are output by default.

OUTACTUAL
outputs the actual values of the solved variables read from the input data set to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTALL
specifies the OUTACTUAL, OUTERRORS, OUTLAGS, OUTPREDICT, and OUTRESID options

OUTERRORS
writes the equation errors to the OUT= data set. These values are normally very close to zero when a simultaneous solution is computed; they can be used to double-check the accuracy of the solution process. It is applicable only if the OUT= option is specified.

OUTLAGS
writes the observations used to start the lags to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTPREDICT
writes the solution values to the OUT= data set. This option is relevant only if the OUT= option is specified.

The OUTPREDICT option is the default unless one of the other output options is used.

OUTRESID
writes the residual values computed as the difference of the solution values and the values for the solution variables read from the input data set to the OUT= data set. This option is applicable only if the OUT= option is specified.

PARMSDATA= *SAS-data-set*
specifies a data set that contains the parameter estimates. See the "Input Data Sets" section for more details.

RESIDDATA= *sas data-set*
specifies a data set that contains the residuals that are to be used in the empirical distribution. This data set can be created using the OUT= option on the Fit statement.

SDATA= *SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The covariance matrix read from the SDATA= data set is used to generate multivariate normal pseudo-random shocks to the equations when the RANDOM= option requests Monte Carlo simulation.

TYPE= *name*
specifies the estimation type. The name specified in the TYPE= option is compared to the _TYPE_ variable in the ESTDATA= and SDATA= data sets to select observations to use in constructing the covariance matrices. When TYPE= is omitted, the last estimation type in the data set is used.

Solution Mode Options: Lag Processing

DYNAMIC

specifies a dynamic solution. In the dynamic solution mode, solved values are used by the lagging functions. DYNAMIC is the default.

NAHEAD= *n*

specifies a simulation of *n*-period-ahead dynamic forecasting. The NAHEAD= option is used to simulate the process of using the model to produce successive forecasts to a fixed forecast horizon, with each forecast using the historical data available at the time the forecast is made.

Note that NAHEAD=1 produces a static (one-step-ahead) solution. NAHEAD=2 produces a solution using one-step-ahead solutions for the first lag (LAG1 functions return static predicted values) and actual values for longer lags. NAHEAD=3 produces a solution using NAHEAD=2 solutions for the first lags, NAHEAD=1 solutions for the second lags, and actual values for longer lags. In general, NAHEAD=*n* solutions use NAHEAD=*n*-1 solutions for LAG1, NAHEAD=*n*-2 solutions for LAG2, and so forth.

START= *s*

specifies static solutions until the *s*th observation and then changes to dynamic solutions. If the START=*s* option is specified, the first observation in the range in which LAG*n* delivers solved predicted values is *s*+*n*, while LAG*n* returns actual values for earlier observations.

STATIC

specifies a static solution. In static solution mode, actual values of the solved variables from the input data set are used by the lagging functions.

Solution Mode Options: Use of Available Data

FORECAST

specifies that the actual value of a solved variable is used as the solution value (instead of the predicted value from the model equations) whenever nonmissing data are available in the input data set. That is, in FORECAST mode, PROC MODEL solves only for those variables that are missing in the input data set.

SIMULATE

specifies that PROC MODEL always solves for all solution variables as a function of the input values of the other variables, even when actual data for some of the solution variables are available in the input data set. SIMULATE is the default.

Solution Mode Options: Numerical Solution Method

JACOBI

computes a simultaneous solution using a Jacobi iteration.

NEWTON

computes a simultaneous solution using Newton's method. When the NEWTON option is selected, the analytic derivatives of the equation errors with respect to the solution variables are computed and memory-efficient sparse matrix techniques are used for factoring the Jacobian matrix.

The NEWTON option can be used to solve both normalized-form and general-form equations and can compute goal-seeking solutions. NEWTON is the default.

SEIDEL
computes a simultaneous solution using a Gauss-Seidel method.

SINGLE
ONEPASS
specifies a single-equation (nonsimultaneous) solution. The model is executed once to compute predicted values for the variables from the actual values of the other endogenous variables. The SINGLE option can only be used for normalized-form equations and cannot be used for goal-seeking solutions.

For more information on these options, see the "Solution Modes" section later in this chapter.

Monte Carlo Simulation Options

PSEUDO= DEFAULT | TWISTER
specifies which pseudo number generator is too be use in generating draws for Monte Carlo simulation. The two pseudo-random number generators supported by the MODEL procedure are a default congruential generator which has period $2^{31} - 1$ and Mersenne-Twister pseudo-random number generator which has an extraordinarily long period $2^{19937} - 1$.

QUASI= NONE|SOBOL|FAURE
specifies a pseudo or quasi-random number generator. Two Quasi-random number generators supported by the MODEL procedure, the Sobol sequence (QUASI=SOBOL) and the Faure sequence (QUASI=FAURE). The default is QUASI=NONE which is the pseudo random number generator.

RANDOM= *n*
repeats the solution *n* times for each BY group, with different random perturbations of the equation errors if the SDATA= option is used; with different random perturbations of the parameters if the ESTDATA= option is used and the ESTDATA= data set contains a parameter covariance matrix; and with different values returned from the random-number generator functions, if any are used in the model program. If RANDOM=0, the random-number generator functions always return zero. See "Monte Carlo Simulation" for details. The default is RANDOM=0.

SEED= *n*
specifies an integer to use as the seed in generating pseudo-random numbers to shock the parameters and equations when the ESTDATA= or the SDATA= options are specified. If *n* is negative or zero, the time of day from the computer's clock is used as the seed. The SEED= option is only relevant if the RANDOM= option is used. The default is SEED=0.

WISHART= *df*
specifies that a Wishart distribution with degrees of freedom *df* be used in place of the normal error covariance matrix. This option is used to model the variance of the error covariance matrix when Monte Carlo simulation is selected.

Options for Controlling the Numerical Solution Process

The following options are useful when you have difficulty converging to the simultaneous solution.

CONVERGE= *value*

specifies the convergence criterion for the simultaneous solution. Convergence of the solution is judged by comparing the CONVERGE= value to the maximum over the equations of

$$\frac{|\epsilon_i|}{|y_i| + 1E-6}$$

if it is computable, otherwise

$$|\epsilon_i|$$

where ϵ_i represents the equation error and y_i represents the solution variable corresponding to the *i*th equation for normalized-form equations. The default is CONVERGE=1E-8.

MAXITER= *n*

specifies the maximum number of iterations allowed for computing the simultaneous solution for any observation. The default is MAXITER=50.

INITIAL= *(variable= [parameter])*

specifies starting values for the parameters

MAXSUBITER= *n*

specifies the maximum number of damping subiterations that are performed in solving a nonlinear system when using the NEWTON solution method. Damping is disabled by setting MAXSUBITER=0. The default is MAXSUBITER=10.

Printing Options

INTGPRINT

prints between data points integration values for the DERT. variables and the auxiliary variables. If you specify the DETAILS option, the integrated derivative variables are printed as well.

ITPRINT

prints the solution approximation and equation errors at each iteration for each observation. This option can produce voluminous output.

PRINTALL

specifies the printing control options DETAILS, ITPRINT, SOLVEPRINT, STATS, and THEIL.

SOLVEPRINT
prints the solution values and residuals at each observation

STATS
prints various summary statistics for the solution values

THEIL
prints tables of Theil inequality coefficients and Theil relative change forecast error measures for the solution values. See "Summary Statistics" in the "Details" section for more information.

Other Options

Other options that can be used on the SOLVE statement include the following that list and analyze the model: BLOCK, GRAPH, LIST, LISTCODE, LISTDEP, LISTDER, and XREF. The LTEBOUND= and MINTIMESTEP= options can be used to control the integration process. The following printing-control options are also available: DETAILS, FLOW, MAXERRORS=, NOPRINT, and TRACE. For complete descriptions of these options, see the PROC MODEL and FIT statement options described earlier in this chapter.

TEST Statement

TEST *["name"] test1 [, test2 ...] [,/ options] ;*

The TEST statement performs tests of nonlinear hypotheses on the model parameters.

The TEST statement applies to the parameters estimated by the associated FIT statement (that is, either the preceding FIT statement or, in the absence of a preceding FIT statement, the following FIT statement). You can specify any number of TEST statements.

If you specify options on the TEST statement, a comma is required before the "/" character separating the test expressions from the options, because the "/" character can also be used within test expressions to indicate division.

Each test is written as an expression optionally followed by an equal sign (=) and a second expression:

[expression] [= expression]

Test expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as "=") and logical operators (such as “&”) cannot be used in TEST statement expressions. Parameters named in test expressions must be among the parameters estimated by the associated FIT statement.

If you specify only one expression in a test, that expression is tested against zero. For example, the following two TEST statements are equivalent:

```
test a + b;

test a + b = 0;
```

When you specify multiple tests on the same TEST statement, a joint test is performed. For example, the following TEST statement tests the joint hypothesis that both A and B are equal to zero.

```
test a, b;
```

To perform separate tests rather than a joint test, use separate TEST statements. For example, the following TEST statements test the two separate hypotheses that A is equal to zero and that B is equal to zero.

```
test a;
test b;
```

You can use the following options in the TEST statement.

WALD
specifies that a Wald test be computed. WALD is the default.

LM
RAO
LAGRANGE
specifies that a Lagrange multiplier test be computed.

LR
LIKE
specifies that a likelihood ratio test be computed.

ALL
requests all three types of tests.

OUT=
specifies the name of an output SAS data set that contains the test results. The format of the OUT= data set produced by the TEST statement is similar to that of the OUTEST= data set produced by the FIT statement.

VAR Statement

VAR *variables [initial_values] ... ;*

The VAR statement declares model variables and optionally provides initial values for the variables' lags. See the "Lag Logic" section for more information.

WEIGHT Statement

WEIGHT *variable;*

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters.

If the weight of an observation is nonpositive, that observation is not used for the estimation. *variable* must be a numeric variable in the input data set.

An alternative weighting method is to use an assignment statement to give values to the special variable _WEIGHT_. The _WEIGHT_ variable must not depend on the parameters being estimated. If both weighting specifications are given, the weights are multiplied together.

Details: Estimation

Estimation Methods

Consider the general nonlinear model:

$$\begin{aligned} \epsilon_t &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \\ \mathbf{z}_t &= Z(\mathbf{x}_t) \end{aligned}$$

where $\mathbf{q} \in R^g$ is a real vector valued function, of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\theta \in R^p$, *g* is the number of equations, *l* is the number of exogenous variables (lagged endogenous variables are considered exogenous here), *p* is the number of parameters and *t* ranges from 1 to *n*. $\mathbf{z}_t \in R^k$ is a vector of instruments. ϵ_t is an unobservable disturbance vector with the following properties:

$$\begin{aligned} E(\epsilon_t) &= 0 \\ E(\epsilon_t \epsilon_t^{'}) &= \Sigma \end{aligned}$$

All of the methods implemented in PROC MODEL aim to minimize an *objective function*. The following table summarizes the objective functions defining the estimators and the corresponding estimator of the covariance of the parameter estimates for each method.

Table 20.1. Summary of PROC MODEL Estimation Methods

Method	Instruments	Objective Function	Covariance of θ
OLS	no	$\mathbf{r}'\mathbf{r}/n$	$(\mathbf{X}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{I})\mathbf{X})^{-1}$
ITOLS	no	$\mathbf{r}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{I})\mathbf{r}/\text{n}$	$(\mathbf{X}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{I})\mathbf{X})^{-1}$
SUR	no	$\mathbf{r}'(\mathbf{S}_{\text{OLS}}^{-1} \otimes \mathbf{I})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1} \otimes \mathbf{I})\mathbf{X})^{-1}$
ITSUR	no	$\mathbf{r}'(\mathbf{S}^{-1} \otimes \mathbf{I})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1} \otimes \mathbf{I})\mathbf{X})^{-1}$
N2SLS	yes	$\mathbf{r}'(\mathbf{I} \otimes \mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{W})\mathbf{X})^{-1}$
IT2SLS	yes	$\mathbf{r}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{W})\mathbf{r}/\text{n}$	$(\mathbf{X}'(\text{diag}(\mathbf{S})^{-1} \otimes \mathbf{W})\mathbf{X})^{-1}$
N3SLS	yes	$\mathbf{r}'(\mathbf{S}_{\text{N2SLS}}^{-1} \otimes \mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1} \otimes \mathbf{W})\mathbf{X})^{-1}$
IT3SLS	yes	$\mathbf{r}'(\mathbf{S}^{-1} \otimes \mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1} \otimes \mathbf{W})\mathbf{X})^{-1}$
GMM	yes	$[n\mathbf{m}_n(\theta)]'\hat{\mathbf{V}}_{\text{N2SLS}}^{-1}[n\mathbf{m}_n(\theta)]/n$	$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$
ITGMM	yes	$[n\mathbf{m}_n(\theta)]'\hat{\mathbf{V}}^{-1}[n\mathbf{m}_n(\theta)]/n$	$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$
FIML	no	$constant + \frac{n}{2}\ln(\det(\mathbf{S}))$ $- \sum_1^n \ln\vert(\mathbf{J}_t)\vert$	$[\hat{\mathbf{Z}}'(\mathbf{S}^{-1} \otimes \mathbf{I})\hat{\mathbf{Z}}]^{-1}$

The column labeled "Instruments" identifies the estimation methods that require instruments. The variables used in this table and the remainder of this chapter are defined as follows:

$n =$ is the number of nonmissing observations.

$g =$ is the number of equations.

$k =$ is the number of instrumental variables.

$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_g \end{bmatrix}$ is the $ng \times 1$ vector of residuals for the *g* equations stacked together.

$\mathbf{r}_i = \begin{bmatrix} q_i(\mathbf{y}_1, \mathbf{x}_1, \theta) \\ q_i(\mathbf{y}_2, \mathbf{x}_2, \theta) \\ \vdots \\ q_i(\mathbf{y}_n, \mathbf{x}_n, \theta) \end{bmatrix}$ is the $n \times 1$ column vector of residuals for the *i*th equation.

S	is a $g \times g$ matrix that estimates Σ, the covariances of the errors across equations (referred to as the **S** matrix).
X	is an $ng \times p$ matrix of partial derivatives of the residual with respect to the parameters.
W	is an $n \times n$ matrix, $\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$.
Z	is an $n \times k$ matrix of instruments.
Y	is a $gk \times ng$ matrix of instruments. $\mathbf{Y} = \mathbf{I}_g \otimes \mathbf{Z}'$.
$\hat{\mathbf{Z}}$	$\hat{\mathbf{Z}} = (\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_p)$ is an $ng \times p$ matrix. $\hat{Z}_i$ is a $ng \times 1$ column vector obtained from stacking the columns of

$$\mathbf{U}\frac{1}{n}\sum_{t=1}^{n}\left(\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial y_t}\right)^{-1}\frac{\partial^2 \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial y_t \partial \theta_i} - \mathbf{Q}_i$$

U	is an $n \times g$ matrix of residual errors. $\mathbf{U} = \epsilon_1, \epsilon_2, \ldots, \epsilon_n{}'$
Q	is the $n \times g$ matrix $\mathbf{q}(\mathbf{y}_1, \mathbf{x}_1, \theta), \mathbf{q}(\mathbf{y}_2, \mathbf{x}_2, \theta), \ldots, \mathbf{q}(\mathbf{y}_n, n, \theta)$.
$\mathbf{Q}_i$	is an $n \times g$ matrix $\frac{\partial \mathbf{Q}}{\partial \theta_i}$.
I	is an $n \times n$ identity matrix.
$\mathbf{J}_t$	is $\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t'}$ which is a $g \times g$ Jacobian matrix.
$\mathbf{m}_n$	is first moment of the crossproduct $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t$. $m_n = \frac{1}{n}\sum_{t=1}^{n} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t$
$\mathbf{z}_t$	is a k column vector of instruments for observation t. $\mathbf{z}_t'$ is also the tth row of **Z**.

$\hat{\mathbf{V}}$	is the $gk \times gk$ matrix representing the variance of the moment functions.
k	is the number of instrumental variables used.
constant	is the constant $\frac{ng}{2}(1 + \ln(2\pi))$.
$\otimes$	is the notation for a Kronecker product.

All vectors are column vectors unless otherwise noted. Other estimates of the covariance matrix for FIML are also available.

Dependent Regressors and Two-Stage Least Squares

Ordinary regression analysis is based on several assumptions. A key assumption is that the independent variables are in fact statistically independent of the unobserved error component of the model. If this assumption is not true–if the regressor varies systematically with the error–then ordinary regression produces inconsistent results. The parameter estimates are *biased*.

Regressors might fail to be independent variables because they are dependent variables in a larger simultaneous system. For this reason, the problem of dependent regressors is often called *simultaneous equation bias*. For example, consider the following two-equation system.

$$y_1 = a_1 + b_1 y_2 + c_1 x_1 + \epsilon_1$$

$$y_2 = a_2 + b_2 y_1 + c_2 x_2 + \epsilon_2$$

In the first equation, y$_2$ is a dependent, or *endogenous,* variable. As shown by the second equation, y$_2$ is a function of y$_1$, which by the first equation is a function of ϵ_1, and therefore y$_2$ depends on ϵ_1. Likewise, y$_1$ depends on ϵ_2 and is a dependent regressor in the second equation. This is an example of a *simultaneous equation* system; y$_1$ and y$_2$ are a function of all the variables in the system.

Using the ordinary least squares (OLS) estimation method to estimate these equations produces biased estimates. One solution to this problem is to replace y$_1$ and y$_2$ on the right-hand side of the equations with predicted values, thus changing the regression problem to the following:

$$y_1 = a_1 + b_1 \hat{y}_2 + c_1 x_1 + \epsilon_1$$

$$y_2 = a_2 + b_2 \hat{y}_1 + c_2 x_2 + \epsilon_2$$

This method requires estimating the predicted values $\hat{y}_1$ and $\hat{y}_2$ through a preliminary, or "first stage," *instrumental regression*. An instrumental regression is a regression of the dependent regressors on a set of *instrumental variables*, which can be any independent variables useful for predicting the dependent regressors. In this example, the equations are linear and the exogenous variables for the whole system are known.

Thus, the best choice for instruments (of the variables in the model) are the variables x_1 and x_2.

This method is known as *two-stage least squares* or 2SLS, or more generally as the *instrumental variables method*. The 2SLS method for linear models is discussed in Pindyck (1981, p. 191-192). For nonlinear models this situation is more complex, but the idea is the same. In nonlinear 2SLS, the derivatives of the model with respect to the parameters are replaced with predicted values. See the section "Choice of Instruments" for further discussion of the use of instrumental variables in nonlinear regression.

To perform nonlinear 2SLS estimation with PROC MODEL, specify the instrumental variables with an INSTRUMENTS statement and specify the 2SLS or N2SLS option on the FIT statement. The following statements show how to estimate the first equation in the preceding example with PROC MODEL.

```
proc model data=in;
   y1 = a1 + b1 * y2 + c1 * x1;
   fit y1 / 2sls;
   instruments x1 x2;
run;
```

The 2SLS or instrumental variables estimator can be computed using a first-stage regression on the instrumental variables as described previously. However, PROC MODEL actually uses the equivalent but computationally more appropriate technique of projecting the regression problem into the linear space defined by the instruments. Thus PROC MODEL does not produce any "first stage" results when you use 2SLS. If you specify the FSRSQ option on the FIT statement, PROC MODEL prints "first-stage R^2" statistic for each parameter estimate.

Formally, the $\hat{\theta}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)\otimes\mathbf{z}_t)\right)'\left(\sum_{t=1}^{n}I\otimes\mathbf{z}_t\mathbf{z}_t'\right)^{-1}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)\otimes\mathbf{z}_t)\right)$$

is the N2SLS estimator of the parameters. The estimate of Σ at the final iteration is used in the covariance of the parameters given in Table 20.1. Refer to Amemiya (1985, p. 250) for details on the properties of nonlinear two-stage least squares.

Seemingly Unrelated Regression

If the regression equations are not simultaneous, so there are no dependent regressors, *seemingly unrelated regression* (SUR) can be used to estimate systems of equations with correlated random errors. The large-sample efficiency of an estimation can be improved if these cross-equation correlations are taken into account. SUR is also known as *joint generalized least squares* or *Zellner regression*. Formally, the $\hat{\theta}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\sum_{t=1}^{n} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'\hat{\Sigma}^{-1}\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$$

is the SUR estimator of the parameters.

The SUR method requires an estimate of the cross-equation covariance matrix, Σ. PROC MODEL first performs an OLS estimation, computes an estimate, $\hat{\Sigma}$, from the OLS residuals, and then performs the SUR estimation based on $\hat{\Sigma}$. The OLS results are not printed unless you specify the OLS option in addition to the SUR option.

You can specify the $\hat{\Sigma}$ to use for SUR by storing the matrix in a SAS data set and naming that data set in the SDATA= option. You can also feed the $\hat{\Sigma}$ computed from the SUR residuals back into the SUR estimation process by specifying the ITSUR option. You can print the estimated covariance matrix $\hat{\Sigma}$ using the COVS option on the FIT statement.

The SUR method requires estimation of the Σ matrix, and this increases the sampling variability of the estimator for small sample sizes. The efficiency gain SUR has over OLS is a large sample property, and you must have a reasonable amount of data to realize this gain. For a more detailed discussion of SUR, refer to Pindyck (1981, p. 331-333).

Three-Stage Least-Squares Estimation

If the equation system is simultaneous, you can combine the 2SLS and SUR methods to take into account both dependent regressors and cross-equation correlation of the errors. This is called *three-stage least squares* (3SLS).

Formally, the $\hat{\theta}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)\otimes\mathbf{z}_t)\right)'\left(\sum_{t=1}^{n}(\hat{\Sigma}\otimes\mathbf{z}_t\mathbf{z}_t')\right)^{-1}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)\otimes\mathbf{z}_t)\right)$$

is the 3SLS estimator of the parameters. For more details on 3SLS, refer to Gallant (1987, p. 435).

Residuals from the 2SLS method are used to estimate the Σ matrix required for 3SLS. The results of the preliminary 2SLS step are not printed unless the 2SLS option is also specified.

To use the three-stage least-squares method, specify an INSTRUMENTS statement and use the 3SLS or N3SLS option on either the PROC MODEL statement or a FIT statement.

Generalized Method of Moments - GMM

For systems of equations with heteroscedastic errors, generalized method of moments (GMM) can be used to obtain efficient estimates of the parameters. See the "Heteroscedasticity" section for alternatives to GMM.

Consider the nonlinear model

$$\begin{aligned} \boldsymbol{\epsilon}_t &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \\ \mathbf{z}_t &= Z(\mathbf{x}_t) \end{aligned}$$

where $\mathbf{z}_t$ is a vector of instruments and ϵ_t is an unobservable disturbance vector that can be serially correlated and nonstationary.

In general, the following orthogonality condition is desired:

$$E(\boldsymbol{\epsilon}_t \otimes \mathbf{z}_t) = 0$$

which states that the expected crossproducts of the unobservable disturbances, $\boldsymbol{\epsilon}_t$, and functions of the observable variables are set to 0. The first moment of the crossproducts is

$$\begin{aligned} \mathbf{m}_n &= \frac{1}{n}\sum_{t=1}^{n} \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) \\ \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t \end{aligned}$$

where $\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) \in R^{gk}$.

The case where $gk > p$ is considered here, where *p* is the number of parameters.

Estimate the true parameter vector θ^0 by the value of $\hat{\theta}$ that minimizes

$$S(\theta, V) = [n\mathbf{m}_n(\theta)]' V^{-1} [n\mathbf{m}_n(\theta)]/n$$

where

$$V = \mathrm{Cov}\left([\mathrm{n}\mathbf{m}_{\mathrm{n}}(\theta^0)], [\mathrm{n}\mathbf{m}_{\mathrm{n}}(\theta^0)]'\right)$$

The parameter vector that minimizes this objective function is the GMM estimator. GMM estimation is requested on the FIT statement with the GMM option.

The variance of the moment functions, V, can be expressed as

$$\begin{aligned} V &= E\left(\sum_{t=1}^{n}\epsilon_t\otimes\mathbf{z}_t\right)\left(\sum_{s=1}^{n}\epsilon_s\otimes\mathbf{z}_s\right)' \\ &= \sum_{t=1}^{n}\sum_{s=1}^{n}E\left[(\epsilon_t\otimes\mathbf{z}_t)(\epsilon_s\otimes\mathbf{z}_s)'\right] \\ &= nS_n^0 \end{aligned}$$

where S_n^0 is estimated as

$$\hat{S}_n = \frac{1}{n}\sum_{t=1}^{n}\sum_{s=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)\otimes\mathbf{z}_t)(\mathbf{q}(\mathbf{y}_s,\mathbf{x}_s,\theta)\otimes\mathbf{z}_s)'$$

Note that $\hat{S}_n$ is a $gk\times gk$ matrix. Because Var $(\hat{S}_n)$ will not decrease with increasing *n* we consider estimators of S_n^0 of the form:

$$\begin{aligned} \hat{S}_n(l(n)) &= \sum_{\tau=-n+1}^{n-1} w\left(\frac{\tau}{l(n)}\right)D\hat{S}_{n,\tau}D \\ \hat{S}_{n,\tau} &= \begin{cases} \sum_{t=1+\tau}^{n}[\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta^{\#})\otimes\mathbf{z}_t][\mathbf{q}(\mathbf{y}_{t-\tau},\mathbf{x}_{t-\tau},\theta^{\#})\otimes\mathbf{z}_{t-\tau}]' & \tau\geq 0 \\ (\hat{S}_{n,-\tau})' & \tau<0 \end{cases} \end{aligned}$$

where $l(n)$ is a scalar function that computes the bandwidth parameter, $w(\cdot)$ is a scalar valued kernel, and the diagonal matrix D is used for a small sample degrees of freedom correction (Gallant 1987). The initial $\theta^{\#}$ used for the estimation of $\hat{S}_n$ is obtained from a 2SLS estimation of the system. The degrees of freedom correction is handled by the VARDEF= option as for the **S** matrix estimation.

The following kernels are supported by PROC MODEL. They are listed with their default bandwidth functions.

Bartlett: KERNEL=BART

$$\begin{aligned} w(x) &= \begin{cases} 1-|x| & |x| <= 1 \\ 0 & \text{otherwise} \end{cases} \\ l(n) &= \frac{1}{2}n^{1/3} \end{aligned}$$

Parzen: KERNEL=PARZEN

$$
\begin{aligned}
w(x) &= \begin{cases} 1-6|x|^2+6|x|^3 & 0 <= |x| <= \frac{1}{2} \\ 2(1-|x|)^3 & \frac{1}{2} <= |x| <= 1 \\ 0 & \text{otherwise} \end{cases} \\
l(n) &= n^{1/5}
\end{aligned}
$$

Quadratic Spectral: KERNEL=QS

$$
\begin{aligned}
w(x) &= \frac{25}{12\pi^2 x^2}\left(\frac{sin(6\pi x/5)}{6\pi x/5} - cos(6\pi x/5)\right) \\
l(n) &= \frac{1}{2}n^{1/5}
\end{aligned}
$$

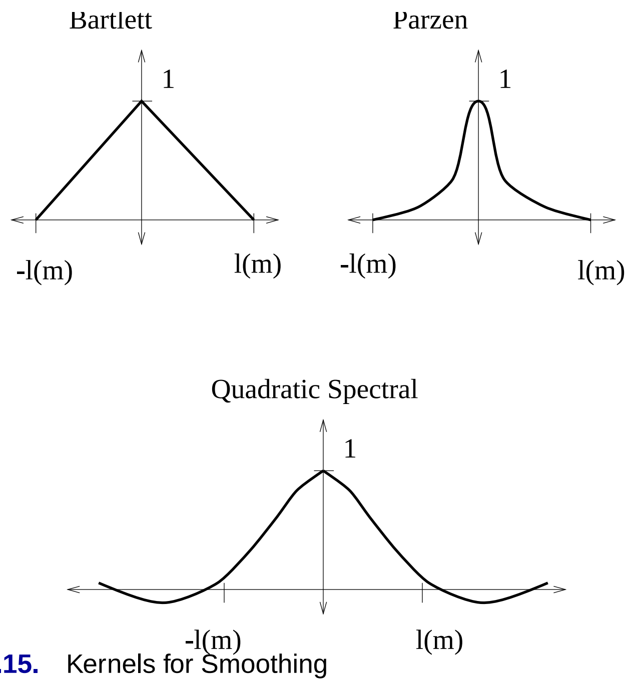


Figure 20.15. Kernels for Smoothing

Details of the properties of these and other kernels are given in Andrews (1991). Kernels are selected with the KERNEL= option; KERNEL=PARZEN is the default. The general form of the KERNEL= option is

```
KERNEL=( PARZEN | QS | BART, c, e )
```

where the $e \geq 0$ and $c \geq 0$ are used to compute the bandwidth parameter as

$$l(n) = cn^e$$

The bias of the standard error estimates increases for large bandwidth parameters. A warning message is produced for bandwidth parameters greater than $n^{\frac{1}{3}}$. For a discussion of the computation of the optimal $l(n)$, refer to Andrews (1991).

The "Newey-West" kernel (Newey (1987)) corresponds to the Bartlett kernel with bandwidth parameter $l(n) = L + 1$. That is, if the "lag length" for the Newey-West kernel is L then the corresponding Model procedure syntax is KERNEL=(bart, L+1, 0).

Andrews (1992) has shown that using prewhitening in combination with GMM can improve confidence interval coverage and reduce over rejection of *t*-statistics at the cost of inflating the variance and MSE of the estimator. Prewhitening can be performed using the %AR macros.

For the special case that the errors are not serially correlated, that is

$$E(e_t \otimes \mathbf{z}_t)(e_s \otimes \mathbf{z}_s) = 0 \qquad t \neq s$$

the estimate for S_n^0 reduces to

$$\hat{S}_n = \frac{1}{n} \sum_{t=1}^{n} [\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t][\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t]'$$

The option KERNEL=(*kernel*,0,) is used to select this type of estimation when using GMM.

Covariance of GMM estimators

The Covariance of GMM estimators given general weighting matrix $\mathbf{V}_{\mathrm{G}}^{-1}$ is

$$[(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})]^{-1}(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}\hat{\mathbf{V}}\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})[(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})]^{-1}$$

By default or when GENGMMV is specified, this is the covariance of GMM estimators.

If the weighting matrix is the same as $\hat{\mathbf{V}}$, then the covariance of GMM estimators becomes

$$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$$

If NOGENGMMV is specified, this is used as the covariance estimators.

Testing Over-Identifying Restrictions

Let *r* be the number of unique instruments times the number of equations. The value *r* represents the number of orthogonality conditions imposed by the GMM method. Under the assumptions of the GMM method, $r - p$ linearly independent combinations of the orthogonality should be close to zero. The GMM estimates are computed by setting these combinations to zero. When *r* exceeds the number of parameters to be estimated, the OBJECTIVE*N, reported at the end of the estimation, is an asymptoticly valid statistic to test the null hypothesis that the over-identifying restrictions of the model are valid. The OBJECTIVE*N is distributed as a chi-square with $r - p$ degrees of freedom (Hansen 1982, p. 1049).

Iterated Generalized Method of Moments - ITGMM

Iterated generalized method of moments is similar to the iterated versions of 2SLS, SUR, and 3SLS. The variance matrix for GMM estimation is re-estimated at each iteration with the parameters determined by the GMM estimation. The iteration terminates when the variance matrix for the equation errors change less than the CONVERGE= value. Iterated generalized method of moments is selected by the ITGMM option on the FIT statement. For some indication of the small sample properties of ITGMM, refer to Ferson and Foerster (1993).

Simulated Method of Moments - SMM

The SMM method uses simulation techniques in model inference and estimation. It is appropriate for estimating models in which integrals appear in the objective function and these integrals can be approximated by simulation. There may be various reasons for integrals to appear in an objective function, for example, transformation of a latent model into an observable model, missing data, random coefficients, heterogeneity, etc.

This simulation method can be used with all the estimation methods except Full Information Maximum Likelihood (FIML) in PROC MODEL. SMM, also known as Simulated Generalized Method of Moments (SGMM), is the default estimation method because of its nice properties.

Estimation Details

A general nonlinear model can be described as

$$\epsilon_t = \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$$

where $\mathbf{q} \in R^g$ is a real vector valued function of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\theta \in R^p$, *g* is the number of equations, *l* is the number of exogenous variables (lagged endogenous variables are considered exogenous here), *p* is the number of parameters, and *t* ranges from 1 to *n*. ϵ_t is an unobservable disturbance vector with the following properties:

$$\begin{aligned} E(\epsilon_t) &= 0 \\ E(\epsilon_t \epsilon_t') &= \Sigma \end{aligned}$$

In many cases it is not possible to write $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$ in a closed form. Instead $\mathbf{q}$ is expressed as an integral of a function $\mathbf{f}$, that is,

$$\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) = \int \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \theta, \mathbf{u}_t) \, dP(\mathbf{u})$$

where $\mathbf{f} \in R^g$ is a real vector valued function of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\theta \in R^p$, and $\mathbf{u}_t \in R^m$, m is the number of stochastic variables with a known distribution $P(\mathbf{u})$. Since the distribution of $\mathbf{u}$ is completely known, it is possible to simulate artificial draws from this distribution. Using such independent draws $\mathbf{u}_{ht}$, $h = 1, \ldots, H$, and the strong law of large numbers, $\mathbf{q}$ can be approximated by

$$\frac{1}{H} \sum_{h=1}^{H} \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \theta, \mathbf{u}_{ht}).$$

Simulated Generalized Method of Moments - SGMM

Generalized Method of Moments (GMM) is widely used to obtain efficient estimates for general model systems. When the moment conditions are not readily available in closed forms but can be approximated by simulation, Simulated Generalized Method of Moments (SGMM) can be used. The SGMM estimators have the nice property of being asymptotically consistent and normally distributed even if the number of draws H is fixed (see McFadden 1989, Pakes and Pollard 1989).

Consider the nonlinear model

$$\begin{aligned} \boldsymbol{\epsilon}_t &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) = \frac{1}{H} \sum_{h=1}^{H} \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \theta, \mathbf{u}_{ht}) \\ \mathbf{z}_t &= Z(\mathbf{x}_t) \end{aligned}$$

where $\mathbf{z}_t \in R^k$ is a vector of k instruments and ϵ_t is an unobservable disturbance vector that can be serially correlated and nonstationary. In case of no instrumental variables, $\mathbf{z}_t$ is 1. $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$ is the vector of moment conditions, and it is approximated by simulation.

In general, theory suggests the following orthogonality condition

$$E(\boldsymbol{\epsilon}_t \otimes \mathbf{z}_t) = 0$$

which states that the expected crossproducts of the unobservable disturbances, $\boldsymbol{\epsilon}_t$, and functions of the observable variables are set to 0. The sample means of the crossproducts are

$$\begin{aligned} \mathbf{m}_n &= \frac{1}{n} \sum_{t=1}^{n} \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) \\ \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \otimes \mathbf{z}_t \end{aligned}$$

where $\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \theta) \in R^{gk}$. The case where $gk > p$, where p is the number of parameters, is considered here. An estimate of the true parameter vector θ^0 is the value of $\hat{\theta}$ that minimizes

$$S(\theta, V) = [n\mathbf{m}_n(\theta)]' V^{-1} [n\mathbf{m}_n(\theta)]/n$$

where

$$V = \mathrm{Cov}\left(\mathbf{m}(\theta^0), \mathbf{m}(\theta^0)'\right).$$

The steps for SGMM are as follows:

1. Start with a positive definite $\hat{V}$ matrix. This $\hat{V}$ matrix can be estimated from a consistent estimator of θ. If $\hat{\theta}$ is a consistent estimator, then $\mathbf{u}_t$ for $t = 1, ..., n$ can be simulated H' number of times. A consistent estimator of V is obtained as

$$\hat{V} = \frac{1}{n}\sum_{t=1}^{n}[\frac{1}{H'}\sum_{h=1}^{H'}\mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \hat{\theta}, \mathbf{u}_{ht})\otimes\mathbf{z}_t][\frac{1}{H'}\sum_{h=1}^{H'}\mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \hat{\theta}, \mathbf{u}_{ht})\otimes\mathbf{z}_t]'$$

H' must be large so that this is an consistent estimator of V.

2. Simulate H number of $\mathbf{u}_t$ for $t = 1, ..., n$. As shown by Gourieroux and Monfort (1993), the number of simulations H does not need to be very large. For $H = 10$, the SGMM estimator achieves 90% of the efficiency of the corresponding GMM estimator. Find $\hat{\theta}$ that minimizes the quadratic product of the moment conditions again with the weight matrix being $\hat{V}^{-1}$.

$$\min_{\theta}[n\mathbf{m}_n(\theta)]'\hat{V}^{-1}[n\mathbf{m}_n(\theta)]/n$$

3. The covariance matrix of $\sqrt{n}\theta$ is given as (Gourieroux and Monfont 1993)

$$\Sigma_1^{-1}D\hat{V}^{-1}V(\hat{\theta})\hat{V}^{-1}D'\Sigma_1^{-1} + \frac{1}{H}\Sigma_1^{-1}D\hat{V}^{-1}E[\mathbf{z}\otimes Var(\mathbf{f}|\mathbf{x})\otimes\mathbf{z}]\hat{V}^{-1}D'\Sigma_1^{-1}$$

where $\Sigma_1 = D\hat{V}^{-1}D$, D is the matrix of partial derivatives of the residuals with respect to the parameters, $V(\hat{\theta})$ is the covariance of moments from estimated parameters $\hat{\theta}$, and $Var(\mathbf{f}|\mathbf{x})$ is the covarince of moments for each observation from simulation. The first term is the variance-covariance matrix of the exact GMM estimator, and the second term accounts for the variation contributed by simulating the moments.

Implementation in PROC MODEL

In PROC MODEL, if the user specifies the GMM and NDRAW options on the FIT statement, PROC MODEL first fits the model using N2SLS and computes $\hat{V}$ using the estimates from N2SLS and H' simulation. If NO2SLS is specified on the FIT statement, $\hat{V}$ is read from VDATA= data set. If the user does not provide a $\hat{V}$ matrix, the initial starting value of θ is used as the estimator for computing the $\hat{V}$ matrix in step 1. If ITGMM option is specified instead of GMM, then PROC MODEL iterates from step 1 to step 3 until the V matrix converges.

The consistency of the parameter estimates is not affected by the variance correction shown in the second term in step 3. The correction on the variance of parameter estimates is not computed by default. To add the adjustment, use ADJSMMV option on the FIT statement. This correction is of the order of $\frac{1}{H}$ and is small even for moderate H.

The following example illustrates how to use SMM to estimate a simple regression model. Suppose the model is

$$y = a + bx + u, u \sim iid\, N(0, s^2).$$

First, consider the problem in GMM context. The first two moments of y are easily derived:

$$\begin{aligned} E(y) &= a + bx \\ E(y^2) &= (a+bx)^2 + s^2 \end{aligned}$$

Rewrite the moment conditions in the form similar to the discussion above:

$$\begin{aligned} \epsilon_{1t} &= y_t - (a + bx_t) \\ \epsilon_{2t} &= y_t^2 - (a+bx_t)^2 - s^2 \end{aligned}$$

Then you can estimate this model using GMM with following code:

```
proc model data=a;
  parms a b s;
  instrument x;
  eq.m1 = y-(a+b*x);
  eq.m2 = y*y - (a+b*x)**2 - s*s;
  bound s > 0;
  fit m1 m2 / gmm;
run;
```

Now suppose you do not have the closed form for the moment conditions. Instead you can simulate the moment conditions by generating H number of simulated samples based on the parameters. Then the simulated moment conditions are

$$\begin{aligned} \epsilon_{1t} &= \frac{1}{H}\sum_{h=1}^{H}\{y_t - (a + bx_t + su_{t,h})\} \\ \epsilon_{2t} &= \frac{1}{H}\sum_{h=1}^{H}\{y_t^2 - (a + bx_t + su_{t,h})^2\} \end{aligned}$$

This model can be estimated using SGMM with the following code:

```
   proc model data=_tmpdata;
          parms a b s;
          instrument x;
          ysim = (a+b*x) + s * rannor( 98711 );
          eq.m1 = y-ysim;
          eq.m2 = y*y - ysim*ysim;
          bound s > 0;
          fit m1 m2 / gmm ndraw=10;
   run;
```

Note that the NDRAW= option tells PROC MODEL that this is a simulation-based estimation. Thus the random number function RANNOR returns random numbers in estimation process. During the simulation, 10 draws of $m1$ and $m2$ are generated for each observation, and the averages enter the objective functions just as the equations specified previously.

Other Estimation Methods

The simulation method can be used not only with GMM and ITGMM, but also with OLS, ITOLS, SUR, ITSUR, N2SLS, IT2SLS, N3SLS, and IT3SLS. These simulation-based methods are similar to the corresponding methods in PROC MODEL; however, the only difference is that the objective functions include the average of the H simulations.

Full Information Maximum Likelihood Estimation - FIML

A different approach to the simultaneous equation bias problem is the full information maximum likelihood (FIML) estimation method (Amemiya 1977).

Compared to the instrumental variables methods (2SLS and 3SLS), the FIML method has these advantages and disadvantages:

- FIML does not require instrumental variables.
- FIML requires that the model include the full equation system, with as many equations as there are endogenous variables. With 2SLS or 3SLS you can estimate some of the equations without specifying the complete system.
- FIML assumes that the equations errors have a multivariate normal distribution. If the errors are not normally distributed, the FIML method may produce poor results. 2SLS and 3SLS do not assume a specific distribution for the errors.
- The FIML method is computationally expensive.

The full information maximum likelihood estimators of θ and σ are the $\hat{\theta}$ and $\hat{\sigma}$ that minimize the negative log likelihood function:

$$\begin{aligned}\mathbf{l}_n(\theta,\sigma) = \; & \tfrac{ng}{2} \ln(2\pi) - \sum_{t=1}^{n} \ln\left(\left|\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t^{'}}\right|\right) + \frac{n}{2}\ln\left(|\Sigma(\sigma)|\right) \\ + \; & \frac{1}{2}\mathrm{tr}\left(\Sigma(\sigma)^{-1}\sum_{\mathrm{t=1}}^{\mathrm{n}} \mathbf{q}(\mathbf{y}_{\mathrm{t}}, \mathbf{x}_{\mathrm{t}}, \theta)\mathbf{q}'(\mathbf{y}_{\mathrm{t}}, \mathbf{x}_{\mathrm{t}}, \theta)\right)\end{aligned}$$

The option FIML requests full information maximum likelihood estimation. If the errors are distributed normally, FIML produces efficient estimators of the parameters. If instrumental variables are not provided the starting values for the estimation are obtained from a SUR estimation. If instrumental variables are provided, then the starting values are obtained from a 3SLS estimation. The negative log likelihood value and the l_2 norm of the gradient of the negative log likelihood function are shown in the estimation summary.

FIML Details

To compute the minimum of $\mathbf{l}_n(\theta, \sigma)$, this function is *concentrated* using the relation

$$\Sigma(\theta) = \frac{1}{n} \sum_{t=1}^{n} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)\mathbf{q}'(\mathbf{y}_t, \mathbf{x}_t, \theta)$$

This results in the concentrated negative log likelihood function:

$$\mathbf{l}_n(\theta) = \frac{ng}{2}(1 + \ln(2\pi)) - \sum_{t=1}^{n} \ln \left| \frac{\partial}{\partial \mathbf{y}_t'} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \right| + \frac{n}{2} \ln |\Sigma(\theta|)$$

The gradient of the negative log likelihood function is

$$\frac{\partial}{\partial \theta_i} \mathbf{l}_n(\theta) = \sum_{t=1}^{n} \nabla_i(t)$$

$$\begin{aligned}
\nabla_i(t) \quad = & \quad -\mathrm{tr}\left(\left(\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t'} \right)^{-1} \frac{\partial^2 \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t' \partial \theta_i} \right) \\
+ & \quad \frac{1}{2} \mathrm{tr}\left(\Sigma(\theta)^{-1} \frac{\partial \Sigma(\theta)}{\partial \theta_i} \right. \\
& \quad \left. \left[I - \Sigma(\theta)^{-1} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)' \right] \right) \\
+ & \quad \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta')\Sigma(\theta)^{-1} \frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \theta_i}
\end{aligned}$$

where

$$\frac{\partial \Sigma(\theta)}{\partial \theta_i} = \frac{2}{n} \sum_{t=1}^{n} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) \frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial \theta_i}$$

The estimator of the variance-covariance of $\hat{\theta}$ (COVB) for FIML can be selected with the COVBEST= option with the following arguments:

CROSS selects the crossproducts estimator of the covariance matrix (default) (Gallant 1987, p. 473):

$$C = \left(\frac{1}{n} \sum_{t=1}^{n} \nabla(t) \nabla'(t) \right)^{-1}$$

where $\nabla(t) = [\nabla_1(t), \nabla_2(t), \ldots, \nabla_p(t)]'$

GLS	selects the generalized least-squares estimator of the covariance matrix. This is computed as (Dagenais 1978) $$C = [\hat{Z}'(\Sigma(\theta)^{-1} \otimes I)\hat{Z}]^{-1}$$ where $\hat{Z} = (\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_p)$ is $ng \times p$ and each $\hat{Z}_i$ column vector is obtained from stacking the columns of $$U \frac{1}{n} \sum_{t=1}^{n} \left(\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial y} \right)^{-1} \frac{\partial^2 \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)'}{\partial \mathbf{y}'_n \partial \theta_i} - Q_i$$ U is an $n \times g$ matrix of residuals and q_i is an $n \times g$ matrix $\frac{\partial \mathbf{Q}}{\partial \theta_i}$.
FDA	selects the inverse of concentrated likelihood Hessian as an estimator of the covariance matrix. The Hessian is computed numerically, so for a large problem this is computationally expensive.

The HESSIAN= option controls which approximation to the Hessian is used in the minimization procedure. Alternate approximations are used to improve convergence and execution time. The choices are as follows.

CROSS	The crossproducts approximation is used.
GLS	The generalized least-squares approximation is used (default).
FDA	The Hessian is computed numerically by finite differences.

HESSIAN=GLS has better convergence properties in general, but COVBEST=CROSS produces the most pessimistic standard error bounds. When the HESSIAN= option is used, the default estimator of the variance-covariance of $\hat{\theta}$ is the inverse of the Hessian selected.

Multivariate t-Distribution Estimation

The multivariate *t*-distribution is specified using the ERRORMODEL statement with the T option. Other method specifications (FIML and OLS, for example) are ignored when the ERRORMODEL statement is used for a distribution other than normal.

The probability density function for the multivariate *t*-distribution is

$$P_q = \frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2}) \, |\Sigma(\sigma)|^{\frac{1}{2}}} * \left(1 + \frac{\mathbf{q}'(\mathbf{y}_t, \mathbf{x}_t, \theta)\Sigma(\sigma)^{-1}\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{df} \right)^{-\frac{df+m}{2}}$$

where m is the number of equations and df is the degrees of freedom.

The maximum likelihood estimators of θ and σ are the $\hat{\theta}$ and $\hat{\sigma}$ that minimize the negative log-likelihood function:

$$\begin{aligned}\mathbf{l}_n(\theta,\sigma) &= -\sum_{t=1}^{n}\ln\left(\frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2})} * \left(1+\frac{q_t'\Sigma^{-1}q_t}{df}\right)^{-\frac{df+m}{2}}\right)\\ &+\frac{n}{2} * \ln\left(|\Sigma|\right) - \sum_{t=1}^{n}\ln\left(\left|\frac{\partial q_t}{\partial y_t'}\right|\right)\end{aligned}$$

The ERRORMODEL statement is used to request the *t*-distribution maximum likelihood estimation. An OLS estimation is done to obtain initial parameter estimates and MSE.*var* estimates. Use NOOLS to turn off this initial estimation. If the errors are distributed normally, *t*-distribution estimation will produce results similar to FIML.

The multivariate model has a single shared degrees of freedom parameter, which is estimated. The degrees of freedom parameter can also be set to a fixed value. The negative log-likelihood value and the l_2 norm of the gradient of the negative log-likelihood function are shown in the estimation summary.

t-Distribution Details

Since a variance term is explicitly specified using the ERRORMODEL statement, $\Sigma(\theta)$ is estimated as a correlation matrix and $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)$ is normalized by the variance. The gradient of the negative log-likelihood function with respect to the degrees of freedom is

$$\begin{aligned}\frac{\partial l_n}{\partial df} &= \frac{nm}{2\,df} - \frac{n}{2}\frac{\Gamma'(\frac{df+m}{2})}{\Gamma(\frac{df+m}{2})} + \frac{n}{2}\frac{\Gamma'(\frac{df}{2})}{\Gamma(\frac{df}{2})} + \\ & 0.5\log(1+\frac{\mathbf{q}'\Sigma^{-1}\mathbf{q}}{df}) - \frac{0.5(df+m)}{(1+\frac{\mathbf{q}'\Sigma^{-1}\mathbf{q}}{df})}\frac{\mathbf{q}'\Sigma^{-1}\mathbf{q}}{df^2}\end{aligned}$$

The gradient of the negative log-likelihood function with respect to the parameters is

$$\frac{\partial l_n}{\partial \theta_i} = \frac{0.5(df+m)}{(1+\mathbf{q}'\Sigma^{-1}\mathbf{q}/df)}\left[\frac{(2\,\mathbf{q}'\Sigma^{-1}\frac{\partial \mathbf{q}}{\partial \theta_i})}{df} + \mathbf{q}'\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_i}\Sigma^{-1}\mathbf{q}\right] - \frac{n}{2}\text{trace}(\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_i})$$

where

$$\frac{\partial\Sigma(\theta)}{\partial\theta_i} = \frac{2}{n}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)\frac{\partial\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)'}{\partial\theta_i}$$

and

$$\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta) = \frac{\epsilon(\theta)}{\sqrt{h(\theta)}} \in R^{m\times n}$$

The estimator of the variance-covariance of $\hat{\theta}$ (COVB) for the *t*-distribution is the inverse of the likelihood Hessian. The gradient is computed analytically and the Hessian is computed numerically.

Empirical Distribution Estimation and Simulation

The following SAS statements fit a model using least squares as the likelihood function, but represent the distribution of the residuals with an empirical CDF. The plot of the empirical probability distribution is shown in the following output.

```
data t;  /* Sum of two normals  */
   format date monyy.;
   do t=0 to 3 by 0.1;
      date = intnx( 'month', '1jun90'd,(t*10)-1);
      y =  0.1 * (rannor(123)-10) +
             .5 *(rannor(456)+10);
      output;
   end;
run;

proc model data=t time=t itprint;
   dependent y;
   parm a 5 ;
   y = a;
   obj = resid.y * resid.y;
   errormodel y ~ general( obj )
      cdf=(empirical=( tails=( t(15) percent= 5)));

   fit y / outns=s out=r;
   id  date;
   solve y / data=t(where=(date='1jun95'd ))
      residdata=r sdata=s random=200 seed=6789 out=monte;
run;

    /*--- Generate the pdf ---*/
proc kde data =monte out=density;
   var y;
run;

symbol1 value=none interpol=join;
proc gplot data=density;
   plot density*y;
run;
```

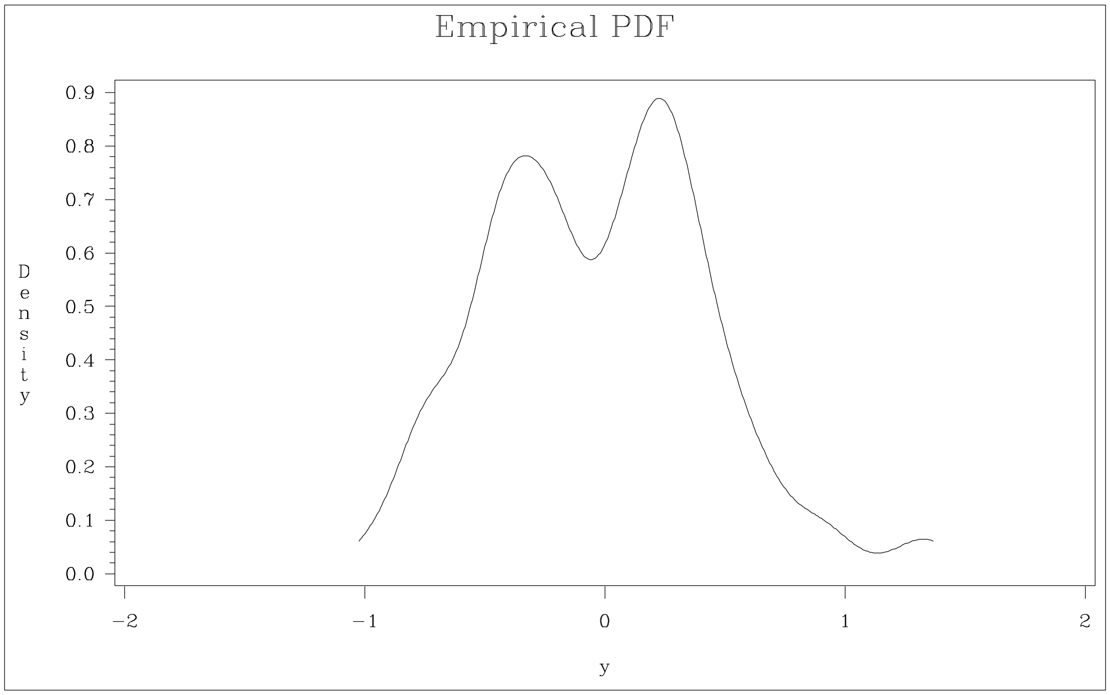


Figure 20.16. Empirical PDF Plot

For simulation, if the CDF for the model is not built in to the procedure, you can use the CDF=EMPIRICAL() option. This uses the sorted residual data to create an empirical CDF. For computing the inverse CDF the program needs to know how to handle the tails. For continuous data, the tail distribution is generally poorly determined. To counter this, the PERCENT= option specifies the percent of the observations to use in constructing each tail. The default for the PERCENT= option is 10.

A normal distribution or a *t*-distribution is used to extrapolate the tails to infinity. The standard errors for this extrapolation are obtained from the data so that the empirical CDF is continuous.

Properties of the Estimates

All of the methods are consistent. Small sample properties may not be good for nonlinear models. The tests and standard errors reported are based on the convergence of the distribution of the estimates to a normal distribution in large samples.

These nonlinear estimation methods reduce to the corresponding linear systems regression methods if the model is linear. If this is the case, PROC MODEL produces the same estimates as PROC SYSLIN.

Except for GMM, the estimation methods assume that the equation errors for each observation are identically and independently distributed with a 0 mean vector and positive definite covariance matrix Σ consistently estimated by **S**. For FIML, the errors need to be normally distributed. There are no other assumptions concerning the distribution of the errors for the other estimation methods.

The consistency of the parameter estimates relies on the assumption that the **S** matrix is a consistent estimate of Σ. These standard error estimates are asymptotically valid, but for nonlinear models they may not be reliable for small samples.

The **S** matrix used for the calculation of the covariance of the parameter estimates is the best estimate available for the estimation method selected. For **S**-iterated methods this is the most recent estimation of Σ. For OLS and 2SLS, an estimate of the **S** matrix is computed from OLS or 2SLS residuals and used for the calculation of the covariance matrix. For a complete list of the **S** matrix used for the calculation of the covariance of the parameter estimates, see Table 20.1.

Missing Values

An observation is excluded from the estimation if any variable used for FIT tasks is missing, if the weight for the observation is not greater than 0 when weights are used, or if a DELETE statement is executed by the model program. Variables used for FIT tasks include the equation errors for each equation, the instruments, if any, and the derivatives of the equation errors with respect to the parameters estimated. Note that variables can become missing as a result of computational errors or calculations with missing values.

The number of usable observations can change when different parameter values are used; some parameter values can be invalid and cause execution errors for some observations. PROC MODEL keeps track of the number of usable and missing observations at each pass through the data, and if the number of missing observations counted during a pass exceeds the number that was obtained using the previous parameter vector, the pass is terminated and the new parameter vector is considered infeasible. PROC MODEL never takes a step that produces more missing observations than the current estimate does.

The values used to compute the Durbin-Watson, R^2, and other statistics of fit are from the observations used in calculating the objective function and do not include any observation for which any needed variable was missing (residuals, derivatives, and instruments).

Details on the Covariance of Equation Errors

There are several **S** matrices that can be involved in the various estimation methods and in forming the estimate of the covariance of parameter estimates. These **S** matrices are estimates of Σ, the true covariance of the equation errors. Apart from the choice of instrumental or noninstrumental methods, many of the methods provided by PROC MODEL differ in the way the various **S** matrices are formed and used.

All of the estimation methods result in a final estimate of Σ, which is included in the output if the COVS option is specified. The final **S** matrix of each method provides the initial **S** matrix for any subsequent estimation.

This estimate of the covariance of equation errors is defined as

$$\mathbf{S} = \mathbf{D}(\mathbf{R}'\mathbf{R})\mathbf{D}$$

where $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_g)$ is composed of the equation residuals computed from the current parameter estimates in an $n \times g$ matrix and **D** is a diagonal matrix that depends on the VARDEF= option.

For VARDEF=N, the diagonal elements of **D** are $1/\sqrt{n}$, where *n* is the number of nonmissing observations. For VARDEF=WGT, *n* is replaced with the sum of the weights. For VARDEF=WDF, *n* is replaced with the sum of the weights minus the model degrees of freedom. For the default VARDEF=DF, the *i*th diagonal element of **D** is $1/\sqrt{n - df_i}$, where df_i is the degrees of freedom (number of parameters) for the *i*th equation. Binkley and Nelson (1984) show the importance of using a degrees-of-freedom correction in estimating Σ. Their results indicate that the DF method produces more accurate confidence intervals for N3SLS parameter estimates in the linear case than the alternative approach they tested. VARDEF=N is always used for the computation of the FIML estimates.

For the fixed **S** methods, the OUTSUSED= option writes the **S** matrix used in the estimation to a data set. This **S** matrix is either the estimate of the covariance of equation errors matrix from the preceding estimation, or a prior Σ estimate read in from a data set when the SDATA= option is specified. For the diagonal **S** methods, all of the off-diagonal elements of the **S** matrix are set to 0 for the estimation of the parameters and for the OUTSUSED= data set, but the output data set produced by the OUTS= option will contain the off-diagonal elements. For the OLS and N2SLS methods, there is no previous estimate of the covariance of equation errors matrix, and the option OUTSUSED= will save an identity matrix unless a prior Σ estimate is supplied by the SDATA= option. For FIML the OUTSUSED= data set contains the **S** matrix computed with VARDEF=N. The OUTS= data set contains the **S** matrix computed with the selected VARDEF= option.

If the COVS option is used, the method is not **S**-iterated, and **S** is not an identity, the OUTSUSED= matrix is included in the printed output.

For the methods that iterate the covariance of equation errors matrix, the **S** matrix is iteratively re-estimated from the residuals produced by the current parameter estimates. This **S** matrix estimate iteratively replaces the previous estimate until both the parameter estimates and the estimate of the covariance of equation errors matrix converge. The final OUTS= matrix and OUTSUSED= matrix are thus identical for the **S**-iterated methods.

Nested Iterations

By default, for **S**-iterated methods, the **S** matrix is held constant until the parameters converge once. Then the **S** matrix is re-estimated. One iteration of the parameter estimation algorithm is performed, and the **S** matrix is again re-estimated. This latter process is repeated until convergence of both the parameters and the **S** matrix. Since the objective of the minimization depends on the **S** matrix, this has the effect of chasing a moving target.

When the NESTIT option is specified, iterations are performed to convergence for the structural parameters with a fixed **S** matrix. The **S** matrix is then re-estimated, the parameter iterations are repeated to convergence, and so on until both the parameters and the **S** matrix converge. This has the effect of fixing the objective function for the inner parameter iterations. It is more reliable, but usually more expensive, to nest the iterations.

R^2

For unrestricted linear models with an intercept successfully estimated by OLS, R^2 is always between 0 and 1. However, nonlinear models do not necessarily encompass the dependent mean as a special case and can produce negative R^2 statistics. Negative R^2's can also be produced even for linear models when an estimation method other than OLS is used and no intercept term is in the model.

R^2 is defined for normalized equations as

$$R^2 = 1 - \frac{SSE}{SSA - \bar{y}^2 \times n}$$

where SSA is the sum of the squares of the actual y's and $\bar{y}$ are the actual means. R^2 cannot be computed for models in general form because of the need for an actual Y.

Minimization Methods

PROC MODEL currently supports two methods for minimizing the objective function. These methods are described in the following sections.

GAUSS

The Gauss-Newton parameter-change vector for a system with *g* equations, *n* nonmissing observations, and *p* unknown parameters is

$$\mathbf{\Delta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r}$$

where Δ is the change vector, **X** is the stacked $ng \times p$ Jacobian matrix of partial derivatives of the residuals with respect to the parameters, and **r** is an $ng \times 1$vector of the stacked residuals. The components of **X** and **r** are weighted by the $\mathbf{S}^{-1}$ matrix. When instrumental methods are used, **X** and **r** are the projections of the Jacobian matrix and residuals vector in the instruments space and not the Jacobian and residuals themselves. In the preceding formula, **S** and W are suppressed. If instrumental variables are used, then the change vector becomes:

$$\mathbf{\Delta} = (\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{r}$$

This vector is computed at the end of each iteration. The objective function is then computed at the changed parameter values at the start of the next iteration. If the objective function is not improved by the change, the Δ vector is reduced by one-half and the objective function is re-evaluated. The change vector will be halved up to MAXSUBITER= times until the objective function is improved.

For FIML the $\mathbf{X}'\mathbf{X}$ matrix is substituted with one of three choices for approximations to the Hessian. See the "FIML Estimation" section in this chapter.

MARQUARDT

The Marquardt-Levenberg parameter change vector is

$$\mathbf{\Delta} = (\mathbf{X}'\mathbf{X} + \lambda \mathrm{diag}(\mathbf{X}'\mathbf{X}))^{-1}\mathbf{X}'\mathbf{r}$$

where Δ is the change vector, and **X** and **r** are the same as for the Gauss-Newton method, described in the preceding section. Before the iterations start, λ is set to a small value (1E-6). At each iteration, the objective function is evaluated at the parameters changed by Δ. If the objective function is not improved, λ is increased to 10λ and the step is tried again. λ can be increased up to MAXSUBITER= times to a maximum of 1E15 (whichever comes first) until the objective function is improved. For the start of the next iteration, λ is reduced to max(λ/10,1E-10).

Convergence Criteria

There are a number of measures that could be used as convergence or stopping criteria. PROC MODEL computes five convergence measures labeled R, S, PPC, RPC, and OBJECT.

When an estimation technique that iterates estimates of Σ is used (that is, IT3SLS), two convergence criteria are used. The termination values can be specified with the CONVERGE=(*p*,*s*) option on the FIT statement. If the second value, *s*, is not specified, it defaults to *p*. The criterion labeled S (given in the following) controls the convergence of the **S** matrix. When S is less than *s*, the **S** matrix has converged. The criterion labeled R is compared to the *p* value to test convergence of the parameters.

The R convergence measure cannot be computed accurately in the special case of singular residuals (when all the residuals are close to 0) or in the case of a 0 objective value. When either the trace of the **S** matrix computed from the current residuals (trace(S)) or the objective value is less than the value of the SINGULAR= option, convergence is assumed.

The various convergence measures are explained in the following:

R is the primary convergence measure for the parameters. It measures the degree to which the residuals are orthogonal to the Jacobian columns, and it approaches 0 as the gradient of the objective function becomes small. R is defined as the square root of

$$\frac{(r'(S^{-1}{\otimes}W)X(X'(S^{-1}{\otimes}W)X)^{-1}X'(S^{-1}{\otimes}W)r)}{(r'(S^{-1}{\otimes}W)r)}$$

where **X** is the Jacobian matrix and **r** is the residuals vector. R is similar to the relative offset orthogonality convergence criterion proposed by Bates and Watts (1981).

In the univariate case, the R measure has several equivalent interpretations:

- the cosine of the angle between the residuals vector and the column space of the Jacobian matrix. When this cosine is 0, the residuals are orthogonal to the partial derivatives of the predicted values with respect to the parameters, and the gradient of the objective function is 0.
- the square root of the R^2 for the current linear pseudo-model in the residuals.
- a norm of the gradient of the objective function, where the norming matrix is proportional to the current estimate of the covariance of the parameter estimates. Thus, using R, convergence is judged when the gradient becomes small in this norm.
- the prospective relative change in the objective function value expected from the next GAUSS step, assuming that the current linearization of the model is a good local approximation.

In the multivariate case, R is somewhat more complicated but is designed to go to 0 as the gradient of the objective becomes small and can still be given the previous interpretations for the aggregation of the equations weighted by $\mathbf{S}^{-1}$.

PPC is the prospective parameter change measure. PPC measures the maximum relative change in the parameters implied by the parameter-change vector computed for the next iteration. At the *k*th iteration, PPC is the maximum over the parameters

$$\frac{|\theta_i^{k+1} - \theta_i^k|}{|\theta|_i^k + 1.0e^{-6}}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k+1} is the prospective value of this parameter after adding the change vector computed for the next iteration. The parameter with the maximum prospective relative change is printed with the value of PPC, unless the PPC is nearly 0.

RPC is the retrospective parameter change measure. RPC measures the maximum relative change in the parameters from the previous iteration. At the *k*th iteration, RPC is the maximum over *i* of

$$\frac{|\theta_i^k - \theta_i^{k-1}|}{|\theta_i^{k-1} + 1.0e^{-6}|}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k-1} is the previous value of this parameter. The name of the parameter with the maximum retrospective relative change is printed with the value of RPC, unless the RPC is nearly 0.

OBJECT measures the relative change in the objective function value between iterations:

$$\frac{|(O^k - O^{k-1}|}{|O^{k-1} + 1.0e^{-6}|}$$

where O^{k-1} is the value of the objective function (O^k) from the previous iteration.

S measures the relative change in the **S** matrix. S is computed as the maximum over *i*, *j* of

$$\frac{|S_{ij}^k - S_{ij}^{k-1}|}{|S_{ij}^{k-1} + 1.0e^{-6}|}$$

where S^{k-1} is the previous **S** matrix. The S measure is relevant only for estimation methods that iterate the **S** matrix.

An example of the convergence criteria output is as follows:

```
                          The MODEL Procedure
                       IT3SLS Estimation Summary

                         Minimization Summary

                   Parameters Estimated            5
                   Method                      Gauss
                   Iterations                     35


                      Final Convergence Criteria

                      R                   0.000883
                      PPC(d1)             0.000644
                      RPC(d1)             0.000815
                      Object               0.00004
                      Trace(S)            3599.982
                      Objective Value     0.435683
                      S                   0.000052
```

Figure 20.17. Convergence Criteria Output

This output indicates the total number of iterations required by the Gauss minimization for all the **S** matrices was 35. The "Trace(S)" is the trace (the sum of the diagonal elements) of the **S** matrix computed from the current residuals. This row is labeled MSE if there is only one equation.

Troubleshooting Convergence Problems

As with any nonlinear estimation routine, there is no guarantee that the estimation will be successful for a given model and data. If the equations are linear with respect to the parameters, the parameter estimates always converge in one iteration. The methods that iterate the **S** matrix must iterate further for the **S** matrix to converge. Nonlinear models may not necessarily converge.

Convergence can be expected only with fully identified parameters, adequate data, and starting values sufficiently close to solution estimates.

Convergence and the rate of convergence may depend primarily on the choice of starting values for the estimates. This does not mean that a great deal of effort should be invested in choosing starting values. First, try the default values. If the estimation fails with these starting values, examine the model and data and re-run the estimation using reasonable starting values. It is usually not necessary that the starting values be very good, just that they not be very bad; choose values that seem plausible for the model and data.

An Example of Requiring Starting Values

Suppose you want to regress a variable Y on a variable X assuming that the variables are related by the following nonlinear equation:

$$y = a + bx^c + \epsilon$$

In this equation, Y is linearly related to a power transformation of X. The unknown parameters are *a*, *b*, and *c*. ϵ is an unobserved random error. Some simulated data was generated using the following SAS statements. In this simulation, $a = 10$, $b = 2$, and the use of the SQRT function corresponds to $c = .5$.

```
data test;
  do i = 1 to 20;
     x = 5 * ranuni(1234);
     y = 10 + 2 * sqrt(x) + .5 * rannor(2345);
     output;
     end;
run;
```

The following statements specify the model and give descriptive labels to the model parameters. Then the FIT statement attempts to estimate *a*, *b*, and *c* using the default starting value .0001.

```
proc model data=test;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y;
run;
```

PROC MODEL prints model summary and estimation problem summary reports and then prints the output shown in Figure 20.18.

```
                              The MODEL Procedure
                                 OLS Estimation

NOTE: The iteration limit is exceeded for OLS.


        ERROR: The parameter estimates failed to converge for OLS after
        100 iterations using CONVERGE=0.001 as the convergence criteria.



                              The MODEL Procedure
                                 OLS Estimation

                                                   N
            Iteration N Obs      R Objective Subit        a        b        c
   OLS            100    20 0.9627    3.9678     2 137.3844 -126.536 -0.00213


                      Gauss Method Parameter Change Vector

                            a              b              c

                    -69367.57       69366.51          -1.16
NOTE: The parameter estimation is abandoned. Check your model and data. If the
      model is correct and the input data are appropriate, try rerunning the
      parameter estimation using different starting values for the parameter
      estimates.
PROC MODEL continues as if the parameter estimates had converged.
```

Figure 20.18. Diagnostics for Convergence Failure

By using the default starting values, PROC MODEL was unable to take even the first step in iterating to the solution. The change in the parameters that the Gauss-Newton method computes is very extreme and makes the objective values worse instead of better. Even when this step is shortened by a factor of a million, the objective function is still worse, and PROC MODEL is unable to estimate the model parameters.

The problem is caused by the starting value of C. Using the default starting value C=.0001, the first iteration attempts to compute better values of A and B by what is, in effect, a linear regression of Y on the 10,000th root of X, which is almost the same as the constant 1. Thus the matrix that is inverted to compute the changes is nearly singular and affects the accuracy of the computed parameter changes.

This is also illustrated by the next part of the output, which displays collinearity diagnostics for the crossproducts matrix of the partial derivatives with respect to the parameters, shown in Figure 20.19.

```
                          The MODEL Procedure
                            OLS Estimation

                       Collinearity Diagnostics

                                 Condition    -----Proportion of Variation----
    Number       Eigenvalue         Number           a           b           c

         1         2.376793         1.0000      0.0000      0.0000      0.0000
         2         0.623207         1.9529      0.0000      0.0000      0.0000
         3     1.684616E-12        1187805      1.0000      1.0000      1.0000
```

Figure 20.19. Collinearity Diagnostics

This output shows that the matrix is singular and that the partials of A, B, and C with respect to the residual are collinear at the point $(0.0001, 0.0001, 0.0001)$ in the parameter space. See the section "Linear Dependencies" for a full explanation of the collinearity diagnostics.

The MODEL procedure next prints the note shown in Figure 20.20, which suggests that you try different starting values.

```
                          The MODEL Procedure
                            OLS Estimation

NOTE: The parameter estimation is abandoned. Check your model and data. If the
      model is correct and the input data are appropriate, try rerunning the
      parameter estimation using different starting values for the parameter
      estimates.
PROC MODEL continues as if the parameter estimates had converged.
```

Figure 20.20. Estimation Failure Note

PROC MODEL then produces the usual printout of results for the nonconverged parameter values. The estimation summary is shown in Figure 20.21. The heading includes the reminder "(Not Converged)."

```
                              The MODEL Procedure
                                OLS Estimation

                            Collinearity Diagnostics

                               Condition    -----Proportion of Variation----
   Number       Eigenvalue        Number            a           b           c

        1         2.376793        1.0000       0.0000      0.0000      0.0000
        2         0.623207        1.9529       0.0000      0.0000      0.0000
        3    1.684616E-12        1187805       1.0000      1.0000      1.0000



                              The MODEL Procedure
                      OLS Estimation Summary (Not Converged)

                              Minimization Summary

                       Parameters Estimated            3
                       Method                      Gauss
                       Iterations                    100
                       Subiterations                 239
                       Average Subiterations        2.39


                          Final Convergence Criteria

                          R                  0.962666
                          PPC(b)             548.1977
                          RPC(b)             540.4224
                          Object             2.633E-6
                          Trace(S)           4.667947
                          Objective Value    3.967755


                             Observations Processed

                                  Read      20
                                  Solved    20
```

Figure 20.21. Nonconverged Estimation Summary

The nonconverged estimation results are shown in Figure 20.22.

```
                                 The MODEL Procedure

                      Nonlinear OLS Summary of Residual Errors
                                   (Not Converged)
                     DF     DF                                                   Adj
  Equation        Model  Error        SSE        MSE  Root MSE  R-Square      R-Sq

  y                   3     17    79.3551     4.6679    2.1605   -1.6812   -1.9966


                 Nonlinear OLS Parameter Estimates (Not Converged)

                                Approx                 Approx
Parameter      Estimate    Std Err    t Value    Pr > |t|   Label

a              137.3844     263342       0.00      0.9996   Intercept
b              -126.536     263342      -0.00      0.9996   Coefficient of
                                                            Transformed X
c              -0.00213     4.4371      -0.00      0.9996   Power Transformation
                                                            Parameter
```

Figure 20.22. Nonconverged Results

Note that the R^2 statistic is negative. An $R^2 < 0$ results when the residual mean square error for the model is larger than the variance of the dependent variable. Negative R^2 statistics may be produced when either the parameter estimates fail to converge correctly, as in this case, or when the correctly estimated model fits the data very poorly.

Controlling Starting Values

To fit the preceding model you must specify a better starting value for C. Avoid starting values of C that are either very large or close to 0. For starting values of A and B, you can either specify values, use the default, or have PROC MODEL fit starting values for them conditional on the starting value for C.

Starting values are specified with the START= option of the FIT statement or on a PARMS statement. For example, the following statements estimate the model parameters using the starting values A=.0001, B=.0001, and C=5.

```
proc model data=test;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=5);
run;
```

Using these starting values, the estimates converge in 16 iterations. The results are shown in Figure 20.23. Note that since the START= option explicitly declares parameters, the parameter C is placed first in the table.

```
                              The MODEL Procedure

                     Nonlinear OLS Summary of Residual Errors

                     DF     DF                                                Adj
  Equation        Model  Error        SSE        MSE   Root MSE  R-Square     R-Sq

  y                   3     17     5.7359     0.3374     0.5809    0.8062   0.7834


                         Nonlinear OLS Parameter Estimates

                                Approx                 Approx
Parameter      Estimate        Std Err    t Value     Pr > |t|    Label

c              0.327079         0.2892       1.13       0.2738    Power Transformation
                                                                  Parameter
a              8.384311         3.3775       2.48       0.0238    Intercept
b              3.505391         3.4858       1.01       0.3287    Coefficient of
                                                                  Transformed X
```

Figure 20.23. Converged Results

Using the STARTITER Option

PROC MODEL can compute starting values for some parameters conditional on starting values you specify for the other parameters. You supply starting values for some parameters and specify the STARTITER option on the FIT statement.

For example, the following statements set C to 1 and compute starting values for A and B by estimating these parameters conditional on the fixed value of C. With C=1 this is equivalent to computing A and B by linear regression on X. A PARMS statement is used to declare the parameters in alphabetical order. The ITPRINT option is used to print the parameter values at each iteration.

```
proc model data=test;
   parms a b c;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=1) / startiter itprint;
run;
```

With better starting values, the estimates converge in only 5 iterations. Counting the 2 iterations required to compute the starting values for A and B, this is 5 fewer than the 12 iterations required without the STARTITER option. The iteration history listing is shown in Figure 20.24.

```
                              The MODEL Procedure
                                OLS Estimation

                                                   N
         Iteration N Obs       R Objective Subit         a         b         c
GRID             0    20 0.9970      161.9     0   0.00010   0.00010   5.00000
GRID             1    20 0.0000     0.9675     0  12.29508   0.00108   5.00000



                                                   N
         Iteration N Obs       R Objective Subit         a         b         c
OLS              0    20 0.6551     0.9675     0  12.29508   0.00108   5.00000
OLS              1    20 0.6882     0.9558     4  12.26426   0.00201   4.44013
OLS              2    20 0.6960     0.9490     4  12.25554   0.00251   4.28262
OLS              3    20 0.7058     0.9428     2  12.24487   0.00323   4.09977
OLS              4    20 0.7177     0.9380     2  12.23186   0.00430   3.89040
OLS              5    20 0.7317     0.9354     2  12.21610   0.00592   3.65450
OLS              6    20 0.7376     0.9289     3  12.20663   0.00715   3.52417
OLS              7    20 0.7445     0.9223     2  12.19502   0.00887   3.37407
OLS              8    20 0.7524     0.9162     2  12.18085   0.01130   3.20393
OLS              9    20 0.7613     0.9106     2  12.16366   0.01477   3.01460
OLS             10    20 0.7705     0.9058     2  12.14298   0.01975   2.80839
OLS             11    20 0.7797     0.9015     2  12.11827   0.02690   2.58933
OLS             12    20 0.7880     0.8971     2  12.08900   0.03712   2.36306
OLS             13    20 0.7947     0.8916     2  12.05460   0.05152   2.13650
OLS             14    20 0.7993     0.8835     2  12.01449   0.07139   1.91695
OLS             15    20 0.8015     0.8717     2  11.96803   0.09808   1.71101
OLS             16    20 0.8013     0.8551     2  11.91459   0.13284   1.52361
OLS             17    20 0.7987     0.8335     2  11.85359   0.17666   1.35745
OLS             18    20 0.8026     0.8311     1  11.71551   0.28373   1.06872
OLS             19    20 0.7945     0.7935     2  11.57666   0.40366   0.89662
OLS             20    20 0.7872     0.7607     1  11.29346   0.65999   0.67059
OLS             21    20 0.7632     0.6885     1  10.81372   1.11483   0.48842
OLS             22    20 0.6976     0.5587     0   9.54889   2.34556   0.30461
OLS             23    20 0.0108     0.2868     0   8.44333   3.44826   0.33232
OLS             24    20 0.0008     0.2868     0   8.39438   3.49500   0.32790


     NOTE: At OLS Iteration 24 CONVERGE=0.001 Criteria Met.
```

Figure 20.24. ITPRINT Listing

The results produced in this case are almost the same as the results shown in Figure 20.23, except that the PARMS statement causes the Parameter Estimates table to be ordered A, B, C instead of C, A, B. They are not exactly the same because the different starting values caused the iterations to converge at a slightly different place. This effect is controlled by changing the convergence criterion with the CONVERGE= option.

By default, the STARTITER option performs one iteration to find starting values for the parameters not given values. In this case the model is linear in A and B, so only one iteration is needed. If A or B were nonlinear, you could specify more than one "starting values" iteration by specifying a number for the STARTITER= option.

Finding Starting Values by Grid Search

PROC MODEL can try various combinations of parameter values and use the combination producing the smallest objective function value as starting values. (For OLS the objective function is the residual mean square.) This is known as a preliminary *grid search*. You can combine the STARTITER option with a grid search.

For example, the following statements try 5 different starting values for C: 10, 5, 2.5, -2.5, -5. For each value of C, values for A and B are estimated. The combination of A, B, and C values producing the smallest residual mean square is then used to start the iterative process.

```
proc model data=test;
   parms a b c;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=10 5 2.5 -2.5 -5) / startiter itprint;
run;
```

The iteration history listing is shown in Figure 20.25. Using the best starting values found by the grid search, the OLS estimation only requires 2 iterations. However, since the grid search required 10 iterations, the total iterations in this case is 12.

```
                              The MODEL Procedure
                                OLS Estimation

                                                   N
         Iteration N Obs       R Objective Subit        a         b         c
GRID             0    20 1.0000    26815.5     0  0.00010  0.00010 10.00000
GRID             1    20 0.0000     1.2193     0 12.51792  0.00000 10.00000
GRID             0    20 0.6012     1.5151     0 12.51792  0.00000  5.00000
GRID             1    20 0.0000     0.9675     0 12.29508  0.00108  5.00000
GRID             0    20 0.7804     1.6091     0 12.29508  0.00108  2.50000
GRID             1    20 0.0000     0.6290     0 11.87327  0.06372  2.50000
GRID             0    20 0.8779     4.1604     0 11.87327  0.06372 -2.50000
GRID             1    20 0.0000     0.9542     0 12.92455 -0.04700 -2.50000
GRID             0    20 0.9998     2776.1     0 12.92455 -0.04700 -5.00000
GRID             1    20 0.0000     1.0450     0 12.86129 -0.00060 -5.00000


                                                   N
         Iteration N Obs       R Objective Subit        a         b         c
OLS              0    20 0.6685     0.6290     0 11.87327  0.06372  2.50000
OLS              1    20 0.6649     0.5871     3 11.79268  0.10083  2.11710
OLS              2    20 0.6713     0.5740     2 11.71445  0.14901  1.81658
OLS              3    20 0.6726     0.5621     2 11.63772  0.20595  1.58705
OLS              4    20 0.6678     0.5471     2 11.56098  0.26987  1.40903
OLS              5    20 0.6587     0.5295     2 11.48317  0.33953  1.26760
OLS              6    20 0.6605     0.5235     1 11.32436  0.48846  1.03784
OLS              7    20 0.6434     0.4997     2 11.18704  0.62475  0.90793
OLS              8    20 0.6294     0.4805     1 10.93520  0.87965  0.73319
OLS              9    20 0.6031     0.4530     1 10.55670  1.26879  0.57385
OLS             10    20 0.6052     0.4526     0  9.62442  2.23114  0.36146
OLS             11    20 0.1652     0.2948     0  8.56683  3.31774  0.32417
OLS             12    20 0.0008     0.2868     0  8.38015  3.50974  0.32664


     NOTE: At OLS Iteration 12 CONVERGE=0.001 Criteria Met.
```

Figure 20.25. ITPRINT Listing

Because no initial values for A or B were provided in the PARAMETERS statement or were read in with a PARMSDATA= or ESTDATA= option, A and B were given the default value of 0.0001 for the first iteration. At the second grid point, C=5, the values of A and B obtained from the previous iterations are used for the initial iteration. If initial values are provided for parameters, the parameters start at those initial values at each grid point.

Guessing Starting Values from the Logic of the Model

Example 20.1, which uses a logistic growth curve model of the U.S. population, illustrates the need for reasonable starting values. This model can be written

$$pop = \frac{a}{1 + \exp(b - c(t - 1790))}$$

where *t* is time in years. The model is estimated using decennial census data of the U.S. population in millions. If this simple but highly nonlinear model is estimated using the default starting values, the estimation fails to converge.

To find reasonable starting values, first consider the meaning of *a* and *c*. Taking the limit as time increases, *a* is the limiting or maximum possible population. So, as a starting value for *a*, several times the most recent population known can be used, for example, one billion (1000 million).

Dividing the time derivative by the function to find the growth rate and taking the limit as *t* moves into the past, you can determine that *c* is the initial growth rate. You can examine the data and compute an estimate of the growth rate for the first few decades, or you can pick a number that sounds like a plausible population growth rate figure, such as 2%.

To find a starting value for *b*, let *t* equal the base year used, 1790, which causes *c* to drop out of the formula for that year, and then solve for the value of *b* that is consistent with the known population in 1790 and with the starting value of *a*. This yields $b = \ln(a/3.9 - 1)$ or about 5.5, where *a* is 1000 and 3.9 is roughly the population for 1790 given in the data. The estimates converge using these starting values.

Convergence Problems

When estimating nonlinear models, you may encounter some of the following convergence problems.

Unable to Improve

The optimization algorithm may be unable to find a step that improves the objective function. If this happens in the Gauss-Newton method, the step size is halved to find a change vector for which the objective improves. In the Marquardt method, λ will be increased to find a change vector for which the objective improves. If, after MAXSUBITER= step-size halvings or increases in λ, the change vector still does not produce a better objective value, the iterations are stopped and an error message is printed.

Failure of the algorithm to improve the objective value can be caused by a CONVERGE= value that is too small. Look at the convergence measures reported at the point of failure. If the estimates appear to be approximately converged, you can accept the NOT CONVERGED results reported, or you can try re-running the FIT task with a larger CONVERGE= value.

If the procedure fails to converge because it is unable to find a change vector that improves the objective value, check your model and data to ensure that all parameters are identified and data values are reasonably scaled. Then, re-run the model with different starting values. Also, consider using the Marquardt method if Gauss-Newton fails; the Gauss-Newton method can get into trouble if the Jacobian matrix is nearly singular or ill-conditioned. Keep in mind that a nonlinear model may be well-identified and well-conditioned for parameter values close to the solution values but unidentified or numerically ill-conditioned for other parameter values. The choice of starting values can make a big difference.

Nonconvergence

The estimates may diverge into areas where the program overflows or the estimates may go into areas where function values are illegal or too badly scaled for accurate calculation. The estimation may also take steps that are too small or that make only marginal improvement in the objective function and, thus, fail to converge within the iteration limit.

When the estimates fail to converge, collinearity diagnostics for the Jacobian crossproducts matrix are printed if there are 20 or fewer parameters estimated. See "Linear Dependencies" later in this section for an explanation of these diagnostics.

Inadequate Convergence Criterion

If convergence is obtained, the resulting estimates will only approximate a minimum point of the objective function. The statistical validity of the results is based on the exact minimization of the objective function, and for nonlinear models the quality of the results depends on the accuracy of the approximation of the minimum. This is controlled by the convergence criterion used.

There are many nonlinear functions for which the objective function is quite flat in a large region around the minimum point so that many quite different parameter vectors may satisfy a weak convergence criterion. By using different starting values, different convergence criteria, or different minimization methods, you can produce very different estimates for such models.

You can guard against this by running the estimation with different starting values and different convergence criteria and checking that the estimates produced are essentially the same. If they are not, use a smaller CONVERGE= value.

Local Minimum

You may have converged to a local minimum rather than a global one. This problem is difficult to detect because the procedure will appear to have succeeded. You can guard against this by running the estimation with different starting values or with a different minimization technique. The START= option can be used to automatically perform a grid search to aid in the search for a global minimum.

Discontinuities

The computational methods assume that the model is a continuous and smooth function of the parameters. If this is not the case, the methods may not work.

If the model equations or their derivatives contain discontinuities, the estimation will usually succeed, provided that the final parameter estimates lie in a continuous interval and that the iterations do not produce parameter values at points of discontinuity or parameter values that try to cross asymptotes.

One common case of discontinuities causing estimation failure is that of an asymptotic discontinuity between the final estimates and the initial values. For example, consider the following model, which is basically linear but is written with one parameter in reciprocal form:

```
   y = a + b * x1 + x2 / c;
```

By placing the parameter C in the denominator, a singularity is introduced into the parameter space at C=0. This is not necessarily a problem, but if the correct estimate of C is negative while the starting value is positive (or vice versa), the asymptotic discontinuity at 0 will lie between the estimate and the starting value. This means that the iterations have to pass through the singularity to get to the correct estimates. The situation is shown in Figure 20.26.

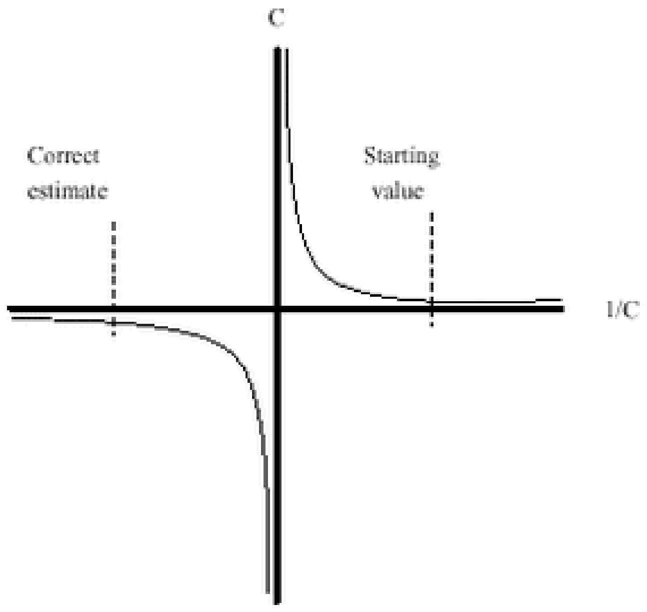


Figure 20.26. Asymptotic Discontinuity

Because of the incorrect sign of the starting value, the C estimate goes off towards positive infinity in a vain effort to get past the asymptote and onto the correct arm of the hyperbola. As the computer is required to work with ever closer approximations to infinity, the numerical calculations break down and an "objective function was not improved" convergence failure message is printed. At this point, the iterations terminate with an extremely large positive value for C. When the sign of the starting value for C is changed, the estimates converge quickly to the correct values.

Linear Dependencies

In some cases, the Jacobian matrix may not be of full rank; parameters may not be fully identified for the current parameter values with the current data. When linear dependencies occur among the derivatives of the model, some parameters appear with a standard error of 0 and with the word BIASED printed in place of the *t* statistic. When this happens, collinearity diagnostics for the Jacobian crossproducts matrix are printed if the DETAILS option is specified and there are twenty or fewer parameters estimated. Collinearity diagnostics are also printed out automatically when a minimization method fails, or when the COLLIN option is specified.

For each parameter, the proportion of the variance of the estimate accounted for by each *principal component* is printed. The principal components are constructed from the eigenvalues and eigenvectors of the correlation matrix (scaled covariance matrix). When collinearity exists, a principal component is associated with proportion of the variance of more than one parameter. The numbers reported are proportions so they will remain between 0 and 1. If two or more parameters have large proportion values associated with the same principle component, then two problems can occur: the computation of the parameter estimates are slow or nonconvergent; and the parameter estimates have inflated variances (Belsley 1980, p. 105-117).

For example, the following cubic model is fit to a quadratic data set:

```
proc model data=test3;
   exogenous x1 ;
   parms b1 a1 c1 ;
   y1 = a1 * x1 + b1 * x1 * x1  + c1 * x1 * x1 *x1;
   fit y1/ collin ;
run;
```

The collinearity diagnostics are shown in Figure 20.27.

```
                                The MODEL Procedure

                             Collinearity Diagnostics

                                Condition    -----Proportion of Variation----
  Number        Eigenvalue         Number          b1          a1          c1

       1          2.942920         1.0000      0.0001      0.0004      0.0002
       2          0.056638         7.2084      0.0001      0.0357      0.0148
       3          0.000442        81.5801      0.9999      0.9639      0.9850
```

Figure 20.27. Collinearity Diagnostics

Notice that the proportions associated with the smallest eigenvalue are almost 1. For this model, removing any of the parameters will decrease the variances of the remaining parameters.

In many models the collinearity might not be clear cut. Collinearity is not necessarily something you remove. A model may need to be reformulated to remove the redundant parameterization or the limitations on the estimatability of the model can be accepted. The GINV=G4 option can be helpful to avoid problems with convergence for models containing collinearities.

Collinearity diagnostics are also useful when an estimation does not converge. The diagnostics provide insight into the numerical problems and can suggest which parameters need better starting values. These diagnostics are based on the approach of Belsley, Kuh, and Welsch (1980).

Iteration History

The options ITPRINT, ITDETAILS, XPX, I, and ITALL specify a detailed listing of each iteration of the minimization process.

ITPRINT Option

The ITPRINT information is selected whenever any iteration information is requested.

The following information is displayed for each iteration:

N	the number of usable observations
Objective	the corrected objective function value
Trace(S)	the trace of the **S** matrix
subit	the number of subiterations required to find a λ or a damping factor that reduces the objective function
R	the R convergence measure

The estimates for the parameters at each iteration are also printed.

ITDETAILS Option

The additional values printed for the ITDETAILS option are:

Theta	is the angle in degrees between Δ, the parameter change vector, and the negative gradient of the objective function.
Phi	is the directional derivative of the objective function in the Δ direction scaled by the objective function.
Stepsize	is the value of the damping factor used to reduce Δ if the Gauss-Newton method is used.
Lambda	is the value of λ if the Marquardt method is used.
Rank(XPX)	If the projected Jacobian crossproducts matrix is singular, the rank of the $\mathbf{X'X}$ matrix is output.

The definitions of PPC and R are explained in the section "Convergence Criteria." When the values of PPC are large, the parameter associated with the criteria is displayed in parentheses after the value.

XPX and I Options

The XPX and the I options select the printing of the augmented $\mathbf{X'X}$ matrix and the augmented $\mathbf{X'X}$ matrix after a *sweep* operation (Goodnight 1979) has been performed on it. An example of the output from the following statements is shown in Figure 20.28.

```
proc model data=test2 ;
   y1 = a1 * x2 * x2 - exp( d1*x1);
   y2 = a2 * x1 * x1 + b2 * exp( d2*x2);
   fit y1 y2 / XPX I ;
run;
```

```
                           The MODEL Procedure
                             OLS Estimation

                Cross Products for System  At OLS Iteration 0

                  a1          d1          a2          b2          d2    Residual

a1           1839468   -33818.35         0.0        0.00    0.000000     3879959
d1            -33818     1276.45         0.0        0.00    0.000000      -76928
a2                 0        0.00     42925.0     1275.15    0.154739      470686
b2                 0        0.00      1275.2       50.01    0.003867       16055
d2                 0        0.00         0.2        0.00    0.000064           2
Residual     3879959   -76928.14    470686.3    16055.07    2.329718    24576144


                  XPX Inverse for System  At OLS Iteration 0

                  a1          d1          a2          b2          d2    Residual

a1          0.000001    0.000028    0.000000      0.0000        0.00           2
d1          0.000028    0.001527    0.000000      0.0000        0.00          -9
a2          0.000000    0.000000    0.000097     -0.0025       -0.08           6
b2          0.000000    0.000000   -0.002455      0.0825        0.95         172
d2          0.000000    0.000000   -0.084915      0.9476    15746.71       11931
Residual    1.952150   -8.546875    5.823969    171.6234    11930.89    10819902
```

Figure 20.28. XPX and I Options Output

The first matrix, labeled "Cross Products," for OLS estimation is

$$\begin{bmatrix} \mathbf{X'X} & \mathbf{X'r} \\ \mathbf{r'X} & \mathbf{r'r} \end{bmatrix}$$

The column labeled "Residual" in the output is the vector $\mathbf{X'r}$, which is the gradient of the objective function. The diagonal scalar value $\mathbf{r'r}$ is the objective function uncorrected for degrees of freedom. The second matrix, labeled "XPX Inverse," is created through a sweep operation on the augmented $\mathbf{X'X}$ matrix to get:

$$\begin{bmatrix} (\mathbf{X'X})^{-1} & (\mathbf{X'X})^{-1}\mathbf{X'r} \\ (\mathbf{X'r})'(\mathbf{X'X})^{-1} & \mathbf{r'r} - (\mathbf{X'r})'(\mathbf{X'X})^{-1}\mathbf{X'r} \end{bmatrix}$$

Note that the residual column is the change vector used to update the parameter estimates at each iteration. The corner scalar element is used to compute the R convergence criteria.

ITALL Option

The ITALL option, in addition to causing the output of all of the preceding options, outputs the **S** matrix, the inverse of the **S** matrix, the CROSS matrix, and the swept CROSS matrix. An example of a portion of the CROSS matrix for the preceding example is shown in Figure 20.29.

```
                              The MODEL Procedure
                                OLS Estimation

                  Crossproducts Matrix  At OLS Iteration 0

                         1     @PRED.y1/@a1     @PRED.y1/@d1     @PRED.y2/@a2

1                    50.00             6409          -239.16           1275.0
@PRED.y1/@a1       6409.08          1839468        -33818.35         187766.1
@PRED.y1/@d1       -239.16           -33818          1276.45          -7253.0
@PRED.y2/@a2       1275.00           187766         -7253.00          42925.0
@PRED.y2/@b2         50.00             6410          -239.19           1275.2
@PRED.y2/@d2          0.00                1            -0.03              0.2
RESID.y1          14699.97          3879959        -76928.14         420582.9
RESID.y2          16052.76          4065028        -85083.68         470686.3

                  Crossproducts Matrix  At OLS Iteration 0

                 @PRED.y2/@b2      @PRED.y2/@d2        RESID.y1        RESID.y2

1                        50.00          0.003803           14700           16053
@PRED.y1/@a1           6409.88          0.813934         3879959         4065028
@PRED.y1/@d1           -239.19         -0.026177          -76928          -85084
@PRED.y2/@a2           1275.15          0.154739          420583          470686
@PRED.y2/@b2             50.01          0.003867           14702           16055
@PRED.y2/@d2              0.00          0.000064               2               2
RESID.y1              14701.77          1.820356        11827102        12234106
RESID.y2              16055.07          2.329718        12234106        12749042
```

Figure 20.29. ITALL Option Cross-Products Matrix Output

Computer Resource Requirements

If you are estimating large systems, you need to be aware of how PROC MODEL uses computer resources such as memory and the CPU so they can be used most efficiently.

Saving Time with Large Data Sets

If your input data set has many observations, the FIT statement does a large number of model program executions. A pass through the data is made at least once for each iteration and the model program is executed once for each observation in each pass. If you refine the starting estimates by using a smaller data set, the final estimation with the full data set may require fewer iterations.

For example, you could use

```
proc model;
   /* Model goes here */
   fit / data=a(obs=25);
   fit / data=a;
```

where OBS=25 selects the first 25 observations in A. The second FIT statement produces the final estimates using the full data set and starting values from the first run.

Fitting the Model in Sections to Save Space and Time

If you have a very large model (with several hundred parameters, for example), the procedure uses considerable space and time. You may be able to save resources by breaking the estimation process into several steps and estimating the parameters in subsets.

You can use the FIT statement to select for estimation only the parameters for selected equations. Do not break the estimation into too many small steps; the total computer time required is minimized by compromising between the number of FIT statements that are executed and the size of the crossproducts matrices that must be processed.

When the parameters are estimated for selected equations, the entire model program must be executed even though only a part of the model program may be needed to compute the residuals for the equations selected for estimation. If the model itself can be broken into sections for estimation (and later combined for simulation and forecasting), then more resources can be saved.

For example, to estimate the following four equation model in two steps, you could use

```
proc model data=a outmodel=part1;
   parms a0-a2 b0-b2 c0-c3 d0-d3;
   y1 = a0 + a1*y2 + a2*x1;
   y2 = b0 + b1*y1 + b2*x2;
   y3 = c0 + c1*y1 + c2*y4 + c3*x3;
   y4 = d0 + d1*y1 + d2*y3 + d3*x4;
   fit y1 y2;
   fit y3 y4;
   fit y1 y2 y3 y4;
run;
```

You should try estimating the model in pieces to save time only if there are more than 14 parameters; the preceding example takes more time, not less, and the difference in memory required is trivial.

Memory Requirements for Parameter Estimation

PROC MODEL is a large program, and it requires much memory. Memory is also required for the SAS System, various data areas, the model program and associated tables and data vectors, and a few crossproducts matrices. For most models, the memory required for PROC MODEL itself is much larger than that required for the model program, and the memory required for the model program is larger than that required for the crossproducts matrices.

The number of bytes needed for two crossproducts matrices, four **S** matrices, and three parameter covariance matrices is

$$8 \times (2 + k + m + g)^2 + 16 \times g^2 + 12 \times (p + 1)^2$$

plus lower-order terms. *m* is the number of unique nonzero derivatives of each residual with respect to each parameter, *g* is the number of equations, *k* is the number of instruments, and *p* is the number of parameters. This formula is for the memory required for 3SLS. If you are using OLS, a reasonable estimate of the memory required for large problems (greater than 100 parameters) is to divide the value obtained from the formula in half.

Consider the following model program.

```
proc model data=test2 details;
   exogenous x1 x2;
   parms b1 100 a1 a2 b2 2.5 c2 55;
   y1 = a1 * y2 + b1 * x1 * x1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2;
   fit y1 y2 / n3sls;
   inst b1 b2 c2 x1 ;
run;
```

The DETAILS option prints the storage requirements information shown in Figure 20.30.

```
                           The MODEL Procedure

                    Storage Requirements for this Problem

                    Order of XPX Matrix                 6
                    Order of S Matrix                   2
                    Order of Cross Matrix              13
                    Total Nonzero Derivatives           5
                    Distinct Variable Derivatives       5
                    Size of Cross matrix              728
```

Figure 20.30. Storage Requirements Information

The matrix $\mathbf{X}'\mathbf{X}$ augmented by the residual vector is called the XPX matrix in the output, and it has the size $m + 1$. The order of the **S** matrix, 2 for this example, is the value of *g*. The CROSS matrix is made up of the *k* unique instruments, a constant column representing the intercept terms, followed by the *m* unique Jacobian variables plus a constant column representing the parameters with constant derivatives, followed by the *g* residuals.

The size of two CROSS matrices in bytes is

$$8 \times (2 + k + m + g)^2 + 2 + k + m + g$$

Note that the CROSS matrix is symmetric, so only the diagonal and the upper triangular part of the matrix is stored. For examples of the CROSS and XPX matrices see "Iteration History" in this section.

The MEMORYUSE Option

The MEMORYUSE option on the FIT, SOLVE, MODEL, or RESET statement may be used to request a comprehensive memory usage summary.

Figure 20.31 shows an example of the output produced by the MEMORYUSE option.

```
                              The MODEL Procedure

                        Memory Usage Summary (in bytes)

                   Symbols                                  5368
                   Strings                                  1057
                   Lists                                    1472
                   Arrays                                     84
                   Statements                                704
                   Opcodes                                   800
                   Parsing                                   640
                   Executable                                220
                   Block option                                0
                   Cross reference                             0
                   Flow analysis                            1024
                   Derivatives                              9406
                   Data vector                               240
                   Cross matrix                              728
                   X'X matrix                                392
                   S matrix                                   96
                   GMM memory                                  0
                   Jacobian                                    0
                   Work vectors                              692
                   Overhead                                 1906
                   -----------------------    --------------
                   Total                                   24829
```

Figure 20.31. MEMORYUSE Option Output for SOLVE Task

Definitions of the memory components follows:

symbols	memory used to store information about variables in the model
strings	memory used to store the variable names and labels
lists	space used to hold lists of variables
arrays	memory used by ARRAY statements
statements	memory used for the list of programming statements in the model
opcodes	memory used to store the code compiled to evaluate the expression in the model program
parsing	memory used in parsing the SAS statements
executable	the compiled model program size (not correct yet)
block option	memory used by the BLOCK option
cross ref.	memory used by the XREF option
flow analysis	memory used to compute the interdependencies of the variables
derivatives	memory used to compute and store the analytical derivatives
data vector	memory used for the program data vector
cross matrix	memory used for one or more copies of the Cross matrix
$\mathbf{X}'\mathbf{X}$ matrix	memory used for one or more copies of the $\mathbf{X}'\mathbf{X}$ matrix
S matrix	memory used for the covariance matrix
GMM memory	additional memory used for the GMM and ITGMM methods
Jacobian	memory used for the Jacobian matrix for SOLVE and FIML
work vectors	memory used for miscellaneous work vectors
overhead	other miscellaneous memory

Testing for Normality

The NORMAL option on the FIT statement performs multivariate and univariate tests of normality.

The three multivariate tests provided are Mardia's skewness test and kurtosis test (Mardia 1980) and the Henze-Zirkler $T_{n,\beta}$ test (Henze and Zirkler 1990). The two univariate tests provided are the Shapiro-Wilk W test and the Kolmogorov-Smirnov test. (For details on the univariate tests, refer to "Tests for Normality" in "The UNIVARIATE Procedure" chapter in the *SAS Procedures Guide*.) The null hypothesis for all these tests is that the residuals are normally distributed.

For a random sample $X_1, \ldots, X_n$, $X_i \in \mathrm{R}^{\mathrm{d}}$, where *d* is the dimension of X_i and *n* is the number of observations, a measure of multivariate skewness is

$$b_{1,d} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} [(X_i - \mu)' S^{-1} (X_j - \mu)]^3$$

where **S** is the sample covariance matrix of **X**. For weighted regression, both **S** and $(X_i - \mu)$ are computed using the weights supplied by the WEIGHT statement or the _WEIGHT_ variable.

Mardia showed that under the null hypothesis $\frac{n}{6} b_{1,d}$ is asymptotically distributed as $\chi^2(d(d+1)(d+2)/6)$.

A measure of multivariate kurtosis is given by

$$b_{2,d} = \frac{1}{n}\sum_{i=1}^{n}[(X_i - \mu)' S^{-1}(X_i - \mu)]^2$$

Mardia showed that under the null hypothesis $b_{2,d}$is asymptotically normally distributed with mean $d(d+2)$ and variance $8d(d+2)/n$.

The Henze-Zirkler test is based on a nonnegative functional $D(.,.)$that measures the distance between two distribution functions and has the property that

$$D(\mathrm{N_d}(0, \mathrm{I_d}), \mathrm{Q}) = 0$$

if and only if

$$Q = \mathrm{N_d}(0, \mathrm{I_d})$$

where $\mathrm{N_d}(\mu, \Sigma_\mathrm{d})$ is a *d*-dimensional normal distribution.

The distance measure $D(.,.)$ can be written as

$$D_\beta(P,Q) = \int_{\mathrm{R^d}} |\hat{P}(t) - \hat{Q}(t)|^2 \varphi_\beta(t)dt$$

where $\hat{P}(t)$ and $\hat{Q}(t)$ are the Fourier transforms of P and Q, and $\varphi_\beta(t)$ is a weight or a kernel function. The density of the normal distribution $\mathrm{N_d}(0, \beta^2\mathrm{I_d})$ is used as $\varphi_\beta(t)$

$$\varphi_\beta(t) = (2\pi\beta^2)^{\frac{-d}{2}}\exp(\frac{-|\mathrm{t}|^2}{2\beta^2}), \quad \mathrm{t} \in \mathrm{R^d}$$

where $|t| = (t' t)^{0.5}$.

The parameter β depends on n as

$$\beta_d(n) = \frac{1}{\sqrt{2}}(\frac{2d+1}{4})^{1/(d+4)}n^{1/(d+4)}$$

The test statistic computed is called $T_\beta(d)$ and is approximately distributed as a log normal. The log normal distribution is used to compute the null hypothesis probability.

$$T_\beta(d) = \frac{1}{n^2} \sum_{j=1}^{n}\sum_{k=1}^{n}\exp(-\frac{\beta^2}{2}|Y_j - Y_k|^2)$$

$$- \quad 2(1+\beta^2)^{-d/2}\frac{1}{n}\sum_{j=1}^{n}\exp(-\frac{\beta^2}{2(1+\beta^2)}|Y_j|^2) + (1+2\beta^2)^{-d/2}$$

where

$$|Y_j - Y_k|^2 = (X_j - X_k)'S^{-1}(X_j - X_k)$$

$$|Y_j|^2 = (X_j - \bar{X})'S^{-1}(X_j - \bar{X})$$

Monte Carlo simulations suggest that $T_\beta(d)$ has good power against distributions with heavy tails.

The Shapiro-Wilk W test is computed only when the number of observations (*n*) is less than 2000.

The following is an example of the output produced by the NORMAL option.

```
                              The MODEL Procedure

                                 Normality Test
              Equation        Test Statistic       Value       Prob

              y1              Shapiro-Wilk W        0.37    <.0001
              y2              Shapiro-Wilk W        0.84    <.0001
              System          Mardia Skewness      286.4    <.0001
                              Mardia Kurtosis      31.28    <.0001
                              Henze-Zirkler T       7.09    <.0001
```

Figure 20.32. Normality Test Output

Heteroscedasticity

One of the key assumptions of regression is that the variance of the errors is constant across observations. If the errors have constant variance, the errors are called *homoscedastic*. Typically, residuals are plotted to assess this assumption. Standard estimation methods are inefficient when the errors are *heteroscedastic* or have non-constant variance.

Heteroscedasticity Tests

The MODEL procedure provides two tests for heteroscedasticity of the errors: White's test and the modified Breusch-Pagan test.

Both White's test and the Breusch-Pagan are based on the residuals of the fitted model. For systems of equations, these tests are computed separately for the residuals of each equation.

The residuals of an estimation are used to investigate the heteroscedasticity of the true disturbances.

The WHITE option tests the null hypothesis

$$H_0 : \sigma_i^2 = \sigma^2 \quad \text{for all i}$$

White's test is general because it makes no assumptions about the form of the heteroscedasticity (White 1980). Because of its generality, White's test may identify specification errors other than heteroscedasticity (Thursby 1982). Thus White's test may be significant when the errors are homoscedastic but the model is misspecified in other ways.

White's test is equivalent to obtaining the error sum of squares for the regression of the squared residuals on a constant and all the unique variables in $\mathbf{J}\otimes\mathbf{J}$, where the matrix **J** is composed of the partial derivatives of the equation residual with respect to the estimated parameters.

Note that White's test in the MODEL procedure is different than White's test in the REG procedure requested by the SPEC option. The SPEC option produces the test from Theorem 2 on page 823 of White (1980). The WHITE option, on the other hand, produces the statistic from Corollary 1 on page 825 of White (1980).

The null hypothesis for the modified Breusch-Pagan test is homosedasticity. The alternate hyposthesis is that the error variance varies with a set of regressors, which are listed in the BREUSCH= option.

Define the matrix Z to be composed of the values of the variables listed in the BREUSCH= option, such that $z_{i,j}$ is the value of the *j*th variable in the BREUSCH= option for the *i*th observation. The null hypothesis of the Breusch-Pagan test is

$$\begin{aligned} \sigma_i^2 &= \sigma^2(\alpha_0 + \boldsymbol{\alpha}'\mathbf{z_i}) \\ H_0 : & \quad \boldsymbol{\alpha} = \mathbf{0} \end{aligned}$$

where σ_i^2 is the error variance for the *i*th observation, and α_0 and $\boldsymbol{\alpha}$ are regression coefficients.

The test statistic for the Breusch-Pagan test is

$$bp = \frac{1}{v}(\mathbf{u} - \bar{u}\mathbf{i})'Z(Z'Z)^{-1}Z'(\mathbf{u} - \bar{u}\mathbf{i})$$

where $\mathbf{u} = (e_1^2, e_2^2, \ldots, e_n^2)$, $\mathbf{i}$ is a $n \times 1$ vector of ones, and

$$v = \frac{1}{n}\sum_{i=1}^{n}(e_i^2 - \frac{\mathbf{e}'\mathbf{e}}{n})^2$$

This is a modified version of the Breusch-Pagan test, which is less sensitive to the assumption of normality than the original test (Greene 1993, p. 395).

The statements in the following example produce the output in Figure 20.33:

```
proc model data=schools;
   parms const inc inc2;

   exp = const + inc * income + inc2 * income * income;
   incsq = income * income;

   fit exp / white breusch=(1 income incsq);
run;
```

```
                              The MODEL Procedure

                            Heteroscedasticity Test
Equation      Test              Statistic   DF  Pr > ChiSq  Variables

exp           White's Test          21.16    4      0.0003  Cross of all vars
              Breusch-Pagan         15.83    2      0.0004  1, income, incsq
```

Figure 20.33. Output for Heteroscedasticity Tests

Correcting for Heteroscedasticity

There are two methods for improving the efficiency of the parameter estimation in the presence of heteroscedastic errors. If the error variance relationships are known, weighted regression can be used or an error model can be estimated. For details on error model estimation see section “Error Covariance Structure Specification”. If the error variance relationship is unknown, GMM estimation can be used.

Weighted Regression

The WEIGHT statement can be used to correct for the heteroscedasticity. Consider the following model, which has a heteroscedastic error term:

$$y_t = 250(e^{-0.2t} - e^{-0.8t}) + \sqrt{(9/t)}\epsilon_t$$

The data for this model is generated with the following SAS statements.

```
data test;
   do t=1 to 25;
      y = 250 * (exp( -0.2 * t ) - exp( -0.8 * t )) +
          sqrt( 9 / t ) * rannor(1);
      output;
   end;
run;
```

If this model is estimated with OLS,

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   fit y;
run;
```

the estimates shown in Figure 20.34 are obtained for the parameters.

```
                              The MODEL Procedure

                       Nonlinear OLS Parameter Estimates

                                         Approx                     Approx
          Parameter        Estimate     Std Err    t Value     Pr > |t|

          b1               0.200977     0.00101     198.60       <.0001
          b2               0.826236     0.00853      96.82       <.0001
```

Figure 20.34. Unweighted OLS Estimates

If both sides of the model equation are multiplied by $\sqrt{t}$, the model will have a homoscedastic error term. This multiplication or weighting is done through the WEIGHT statement. The WEIGHT statement variable operates on the squared residuals as

$$\epsilon_t^{'}\epsilon_t = weight \times \mathbf{q}_t^{'}\mathbf{q}_t$$

so that the WEIGHT statement variable represents the square of the model multiplier. The following PROC MODEL statements corrects the heteroscedasticity with a WEIGHT statement

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   fit y;
   weight t;
run;
```

Note that the WEIGHT statement follows the FIT statement. The weighted estimates are shown in Figure 20.35.

```
                              The MODEL Procedure

                       Nonlinear OLS Parameter Estimates

                                         Approx                     Approx
          Parameter        Estimate     Std Err    t Value     Pr > |t|

          b1               0.200503    0.000844     237.53       <.0001
          b2               0.816701      0.0139      58.71       <.0001
```

Figure 20.35. Weighted OLS Estimates

The weighted OLS estimates are identical to the output produced by the following PROC MODEL example:

```
proc model data=test;
   parms b1 0.1 b2 0.9;
```

```
      y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
      _weight_ = t;
      fit y;
   run;
```

If the WEIGHT statement is used in conjunction with the _WEIGHT_ variable, the two values are multiplied together to obtain the weight used.

The WEIGHT statement and the _WEIGHT_ variable operate on all the residuals in a system of equations. If a subset of the equations needs to be weighted, the residuals for each equation can be modified through the RESID. variable for each equation. The following example demonstrates the use of the RESID. variable to make a homoscedastic error term:

```
   proc model data=test;
      parms b1 0.1 b2 0.9;
      y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
      resid.y = resid.y * sqrt(t);
      fit y;
   run;
```

These statements produce estimates of the parameters and standard errors that are identical to the weighted OLS estimates. The reassignment of the RESID.Y variable must be done after Y is assigned, otherwise it would have no effect. Also, note that the residual (RESID.Y) is multiplied by $\sqrt{t}$. Here the multiplier is acting on the residual before it is squared.

GMM Estimation

If the form of the heteroscedasticity is unknown, generalized method of moments estimation (GMM) can be used. The following PROC MODEL statements use GMM to estimate the example model used in the preceding section:

```
   proc model data=test;
      parms b1 0.1 b2 0.9;
      y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
      fit y / gmm;
      instruments b1 b2;
   run;
```

GMM is an instrumental method, so instrument variables must be provided.

GMM estimation generates estimates for the parameters shown in Figure 20.36.

```
                              The MODEL Procedure

                        Nonlinear GMM Parameter Estimates

                                           Approx                       Approx
          Parameter       Estimate        Std Err     t Value        Pr > |t|

          b1              0.200487       0.000807      248.38          <.0001
          b2              0.822148         0.0142       57.95          <.0001
```

Figure 20.36. GMM Estimation for Heteroscedasticity

Heteroscedasticity-Consistent Covariance Matrix Estimation

Homoscedasticity is required for ordinary least-squares regression estimates to be efficient. A nonconstant error variance, heteroscedasticity, causes the OLS estimates to be inefficient, and the usual OLS covariance matrix, $\hat{\Sigma}$, is generally invalid.

$$\hat{\Sigma} = \sigma^2(X'X)^{-1}$$

When the variance of the errors of a classical linear model

$$Y = X\beta + \epsilon$$

is not constant across observations (heteroscedastic), so that $\sigma_i^2 \neq \sigma_j^2$ for some $j > 1$, the OLS estimator

$$\hat{\beta}_{OLS} = (X'X)^{-1}X'Y$$

is unbiased but it is inefficient. Models that take into account the changing variance can make more efficient use of the data. When the variances, σ_t^2, are known, generalized least squares (GLS) can be used and the estimator

$$\hat{\beta}_{GLS} = (X'\Omega X)^{-1}X'\Omega^{-1}Y$$

where

$$\Omega = \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_T^2 \end{bmatrix}$$

is unbiased and efficient. However, GLS is unavailable when the variances, σ_t^2, are unknown.

To solve this problem White (1980) proposed a heteroscedastic consistent-covariance matrix estimator (HCCME)

$$\hat{\Sigma} = (X'X)^{-1}X'\hat{\Omega}X(X'X)^{-1}$$

that is consistent as well as unbiased, where

$$\hat{\Omega}_0 = \begin{bmatrix} \epsilon_1^2 & 0 & 0 & 0 \\ 0 & \epsilon_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \epsilon_T^2 \end{bmatrix}$$

and $\epsilon_t = Y_t - X_t \hat{\beta_{OLS}}$.

This estimator is considered somewhat unreliable in finite samples. Therefore, Davidson and MacKinnon (1993) propose three different modifications to estimating $\hat{\Omega}$. The first solution is to simply multiply ϵ_t^2 by $\frac{n}{n-df}$, where *n* is the number of observations and *df* is the number of explanatory variables, so that

$$\hat{\Omega}_1 = \begin{bmatrix} \frac{n}{n-df}\,\epsilon_1^2 & 0 & 0 & 0 \\ 0 & \frac{n}{n-df}\,\epsilon_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{n}{n-df}\,\epsilon_n^2 \end{bmatrix}$$

The second solution is to define

$$\hat{\Omega}_2 = \begin{bmatrix} \frac{\epsilon_1^2}{1-\hat{h}_1} & 0 & 0 & 0 \\ 0 & \frac{\epsilon_2^2}{1-\hat{h}_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{\epsilon_n^2}{1-\hat{h}_n} \end{bmatrix}$$

where $\hat{h}_t = X_t(X'X)^{-1}X_t'$.

The third solution, called the "jackknife," is to define

$$\hat{\Omega}_3 = \begin{bmatrix} \frac{\epsilon_1^2}{(1-\hat{h}_1)^2} & 0 & 0 & 0 \\ 0 & \frac{\epsilon_2^2}{(1-\hat{h}_2)^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{\epsilon_n^2}{(1-\hat{h}_T)^2} \end{bmatrix}$$

MacKinnon and White (1985) investigated these three modified HCCMEs, including the original HCCME, based on finite-sample performance of pseudo-t statistics. The original HCCME performed the worst. The first modification performed better. The second modification performed even better than the first, and the third modification performed the best. They concluded that the original HCCME should never be used in finite sample estimation, and that the second and third modifications should be used over the first modification if the diagonals of $\hat{\Omega}$ are available.

Seemingly Unrelated Regression HCCME

Extending the discussion to systems of g equations, the HCCME for SUR estimation is

$$(\tilde{X}'\tilde{X})^{-1}\tilde{X}'\hat{\Omega}\tilde{X}(\tilde{X}'\tilde{X})^{-1}$$

where $\tilde{X}$ is a $n\,g \times k$ matrix with the first g rows representing the first observation, the next g rows representing the second observation, and so on. $\hat{\Omega}$ is now a $n\,g \times n\,g$ block diagonal matrix with typical block $g \times g$

$$\hat{\Omega}_i = \begin{bmatrix} \psi_{1,i}\,\psi_{1,i} & \psi_{1,i}\,\psi_{2,i} & \ldots & \psi_{1,i}\,\psi_{g,i} \\ \psi_{2,i}\,\psi_{1,i} & \psi_{2,i}\,\psi_{2,i} & \ldots & \psi_{2,i}\,\psi_{g,i} \\ \vdots & \vdots & \vdots & \vdots \\ \psi_{g,i}\,\psi_{1,i} & \psi_{g,i}\,\psi_{2,i} & \ldots & \psi_{g,i}\,\psi_{g,i} \end{bmatrix}$$

where

$$\psi_{j,i} = \epsilon_{j,i} \quad HC_0$$

or

$$\psi_{j,i} = \sqrt{\frac{n}{n - df}}\epsilon_{j,i} \quad HC_1$$

or

$$\psi_{j,i} = \epsilon_{j,i}/\sqrt{1 - \hat{h}_i} \quad HC_2$$

or

$$\psi_{j,i} = \epsilon_{j,i}/(1 - \hat{h}_i) \quad HC_3$$

Two- and Three-Stage Least Squares HCCME

For two- and three-stage least squares, the HCCME for a g equation system is

$$CovF(\hat{\Omega})Cov$$

where

$$Cov = \left(\frac{1}{n}X'(I \otimes Z(Z'Z)^{-1}Z')X\right)^{-1}$$

is the normal covariance matrix without the S matrix and

$$F(\Omega) = \frac{1}{n}\sum_{i}^{g}\sum_{j}^{g} X_i'Z(Z'Z)^{-1}Z'\hat{\Omega}_{ij}Z(Z'Z)^{-1}Z'X_j$$

where X_j is a $n \times p$ matrix with the jth equations regressors in the appropriate columns and zeros everywhere else.

$$\hat{\Omega}_{ij} = \begin{bmatrix} \psi_{i,1}\psi_{j,1} & 0 & 0 & 0 \\ 0 & \psi_{i,2}\psi_{j,2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \psi_{i,n}\psi_{j,n} \end{bmatrix}$$

For 2SLS $\hat{\Omega}_{ij} = 0$ when $i \neq j$. The ϵ_t used in $\hat{\Omega}$ is computed using the parameter estimates obtained from the instrumental variables estimation.

The leverage value for the ith equation used in the HCCME=2 and HCCME=3 methods is computed as conditional on the first stage as

$$h_{ti} = Z_t(Z'Z)^{-1}X_i(X'(I \otimes Z(Z`*Z)^{-1}Z`)X)^{-1}X_i'Z(Z'Z)^{-1}Z_t'$$

for 2SLS and

$$h_{ti} = Z_t(Z'Z)^{-1}X_i(X'(S^{-1} \otimes Z(Z`*Z)^{-1}Z`)X)^{-1}X_i'Z(Z'Z)^{-1}Z_t'/S_{ii}$$

for 3SLS.

Testing for Autocorrelation

The GODFREY= option on the FIT statement produces the Godfrey Lagrange multiplier test for serially correlated residuals for each equation (Godfrey 1978a and 1978b). n is the maximum autoregressive order, and specifies that Godfrey's tests be computed for lags 1 through n. The default number of lags is four.

The tests are performed separately for each equation estimated by the FIT statement. When a nonlinear model is estimated, the test is computed using a linearized model.

The following is an example of the output produced by the GODFREY=3 option:

```
                          Godfrey Test Output

                          The MODEL Procedure

                   Godfrey's Serial Correlation Test

           Equation        Alternative        LM     Pr > LM

           y                    1            6.63     0.0100
                                2            6.89     0.0319
                                3            6.96     0.0732
```

Figure 20.37. Autocorrelation Test Output

The three variations of the test reported by the GODFREY=3 option are designed to have power against different alternative hypothesis. Thus, if the residuals in fact have only first-order autocorrelation, the lag 1 test will have the most power for rejecting the null hypothesis of uncorrelated residuals. If the residuals have second- but not higher-order autocorrelation, the lag 2 test may be more likely to reject; the same is true for third-order autocorrelation and the lag 3 test.

The null hypothesis of Godfrey's tests is that the equation residuals are white noise. However, if the equation includes autoregressive error model of order p (AR(p),) then the lag i test, when considered in terms of the structural error, is for the null hypothesis that the structural errors are from an AR(p) process versus the alternative hypothesis that the errors are from an AR($p + i$) process.

The alternative ARMA(p, i) process is locally equivalent to the alternative AR($p + i$) process with respect to the null model AR(p). Thus, the GODFREY= option results are also a test of AR(p) errors against the alternative hypothesis of ARMA(p, i) errors. Refer to Godfrey (1978a and 1978b) for more detailed information.

Transformation of Error Terms

In PROC MODEL you can control the form of the error term. By default the error term is assumed to be additive. This section demonstrates how to specify nonadditive error terms and discusses the effects of these transformations.

Models with Nonadditive Errors

The estimation methods used by PROC MODEL assume that the error terms of the equations are independently and identically distributed with zero means and finite variances. Furthermore, the methods assume that the RESID.*name* equation variable for normalized form equations or the EQ.*name* equation variable for general form equations contains an estimate of the error term of the true stochastic model whose parameters are being estimated. Details on RESID.*name* and EQ.*name* equation variables are in the section "Model Translations."

To illustrate these points, consider the common loglinear model

$$y = \alpha x^{\beta} \tag{1}$$

$$\ln y = a + b\ln(x) \tag{2}$$

where *a*=log(α) and *b*=β. Equation (2) is called the *log form* of the equation in contrast to equation (1), which is called the *level form* of the equation. Using the SYSLIN procedure, you can estimate equation (2) by specifying

```
proc syslin data=in;
   model logy=logx;
run;
```

where LOGY and LOGX are the logs of Y and X computed in a preceding DATA step. The resulting values for INTERCEPT and LOGX correspond to *a* and *b* in equation (2).

Using the MODEL procedure, you can try to estimate the parameters in the level form (and avoid the DATA step) by specifying

```
proc model data=in;
   parms alpha beta;
   y = alpha * x ** beta;
   fit y;
run;
```

where ALPHA and BETA are the parameters in equation (1).

Unfortunately, at least one of the preceding is wrong; an ambiguity results because equations (1) and (2) contain no explicit error term. The SYSLIN and MODEL procedures both deal with additive errors; the residual used (the estimate of the error term in the equation) is the difference between the predicted and actual values (of LOGY for PROC SYSLIN and of Y for PROC MODEL in this example). If you perform the regressions discussed previously, PROC SYSLIN estimates equation (3) while PROC MODEL estimates equation (4).

$$\ln y = a + b\ln(x) + \epsilon \qquad (3)$$

$$y = \alpha x^{\beta} + \xi \qquad (4)$$

These are different statistical models. Equation (3) is the log form of equation (5)

$$y = \alpha x^{\beta}\mu \qquad (5)$$

where μ=e$^{\epsilon}$. Equation (4), on the other hand, cannot be linearized because the error term ξ (different from μ) is additive in the level form.

You must decide whether your model is equation (4) or (5). If the model is equation (4), you should use PROC MODEL. If you linearize equation (1) without considering the error term and apply SYSLIN to MODEL LOGY=LOGX, the results will be wrong. On the other hand, if your model is equation (5) (in practice it usually is), and you want to use PROC MODEL to estimate the parameters in the *level* form, you must do something to account for the multiplicative error.

PROC MODEL estimates parameters by minimizing an objective function. The objective function is computed using either the RESID.-prefixed equation variable or the EQ.-prefixed equation variable. You must make sure that these prefixed equation variables are assigned an appropriate error term. If the model has additive errors that satisfy the assumptions, nothing needs to be done. In the case of equation (5), the error is nonadditive and the equation is in normalized form, so you must alter the value of RESID.Y.

The following assigns a valid estimate of μ to RESID.Y:

```
y = alpha * x ** beta;
resid.y = actual.y / pred.y;
```

However, μ=e$^{\epsilon}$ and, therefore, μ cannot have a mean of zero and you cannot consistently estimate α and β by minimizing the sum of squares of an estimate of μ. Instead, you use $\epsilon = \ln\mu$.

```
proc model data=in;
   parms alpha beta;
   y = alpha * x ** beta;
   resid.y = log( actual.y / pred.y );
   fit y;
run;
```

If the model was expressed in general form, this transformation becomes

```
proc model data=in;
   parms alpha beta;
   EQ.trans = log( y / (alpha * x ** beta));
   fit trans;
run;
```

Both examples produce estimates of α and β of the level form that match the estimates of *a* and *b* of the log form. That is, ALPHA=exp(INTERCEPT) and BETA=LOGX, where INTERCEPT and LOGX are the PROC SYSLIN parameter estimates from the MODEL LOGY=LOGX. The standard error reported for ALPHA is different from that for the INTERCEPT in the log form.

The preceding example is not intended to suggest that loglinear models should be estimated in level form but, rather, to make the following points:

- Nonlinear transformations of equations involve the error term of the equation, and this should be taken into account when transforming models.
- The RESID.-prefixed and the EQ.-prefixed equation variables for models estimated by the MODEL procedure must represent additive errors with zero means.
- You can use assignments to RESID.-prefixed and EQ.-prefixed equation variables to transform error terms.
- Some models do not have additive errors or zero means, and many such models can be estimated using the MODEL procedure. The preceding approach applies not only to multiplicative models but to any model that can be manipulated to isolate the error term.

Predicted Values of Transformed Models

Nonadditive or transformed errors affect the distribution of the predicted values, as well as the estimates. For the preceding loglinear example, the MODEL procedure produces consistent parameter estimates. However, the predicted values for Y computed by PROC MODEL are not unbiased estimates of the expected values of Y, although they do estimate the conditional median Y values.

In general, the predicted values produced for a model with nonadditive errors are not unbiased estimates of the conditional means of the endogenous value. If the model can be transformed to a model with additive errors by using a *monotonic* transformation, the predicted values estimate the conditional medians of the endogenous variable.

For transformed models in which the biasing factor is known, you can use programming statements to correct for the bias in the predicted values as estimates of the endogenous means. In the preceding loglinear case, the predicted values will be biased by the factor exp(σ^2/2). You can produce approximately unbiased predicted values in this case by writing the model as

```
proc model data=in;
   parms alpha beta;
   y=alpha * x ** beta;
   resid.y = log( actual.y / pred.y );

   fit y;
run;
```

Refer to Miller (1984) for a discussion of bias factors for predicted values of transformed models.

Note that models with transformed errors are not appropriate for Monte Carlo simulation using the SDATA= option. PROC MODEL computes the OUTS= matrix from the transformed RESID.-prefixed equation variables, while it uses the SDATA= matrix to generate multivariate normal errors, which are added to the predicted values. This method of computing errors is inconsistent when the equation variables have been transformed.

Error Covariance Structure Specification

One of the key assumptions of regression is that the variance of the errors is constant across observations. Correcting for heteroscedasticity improves the efficiency of the estimates.

Consider the following general form for models:

$$\begin{aligned} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \varepsilon_{\mathbf{t}} \\ \varepsilon_{\mathbf{t}} &= H_t * \epsilon_{\mathbf{t}} \end{aligned}$$

$$
\begin{aligned}
H_t &= \begin{bmatrix} \sqrt{h_{t,1}} & 0 & \dots & 0 \\ 0 & \sqrt{h_{t,2}} & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \sqrt{h_{t,g}} \end{bmatrix} \\
\mathbf{h_t} &= \mathbf{g}(\mathbf{y}_t, \mathbf{x}_t, \phi)
\end{aligned}
$$

where $\epsilon_{\mathbf{t}} \sim N(0, \Sigma)$.

For models that are homoscedastic,

$$h_t = 1$$

If you had a model which was heteroscedastic with known form you can improve the efficiency of the estimates by performing a weighted regression. The weight variable, using this notation, would be $1/\sqrt{h_t}$.

If the errors for a model are heteroscedastic and the functional form of the variance is known, the model for the variance can be estimated along with the regression function.

To specify a functional form for the variance, assign the function to an H.*var* variable where *var* is the equation variable. For example, if you wanted to estimate the scale parameter for the variance of a simple regression model

$$y = a * x + b$$

you can specify

```
proc model data=s;
   y = a * x + b;
   h.y = sigma**2;
fit y;
```

Consider the same model with the following functional form for the variance:

$$h_t = \sigma^2 * x^{2*\alpha}$$

This would be written as

```
proc model data=s;
   y = a * x + b;
   h.y = sigma**2 * x**(2*alpha);
fit y;
```

There are three ways to model the variance in the MODEL procedure; Feasible generalized least squares; Generalized method of moments; and Full information maximum likelihood.

Feasible GLS

A simple approach to estimating a variance function is to estimate the mean parameters θ using some auxiliary method, such as OLS, and then use the residuals of that estimation to estimate the parameters ϕ of the variance function. This scheme is called *feasible GLS*. It is possible to use the residuals from an auxiliary method for the purpose of estimating ϕ because in many cases the residuals consistently estimate the error terms.

For all estimation methods except GMM and FIML, using the H.var syntax specifies that feasible GLS will be used in the estimation. For feasible GLS the mean function is estimated by the usual method. The variance function is then estimated using pseudolikelihood (PL) function of the generated residuals. The objective function for the PL estimation is

$$p_n(\sigma, \theta) = \sum_{i=1}^{n} \left(\frac{(y_i - f(x_i, \hat{\beta}))^2}{\sigma^2 h(z_i, \theta)} + \log[\sigma^2 h(z_i, \theta)] \right)$$

Once the variance function has been estimated the mean function is re-estimated using the variance function as weights. If an S-iterated method is selected, this process is repeated until convergence (iterated feasible GLS).

Note, feasible GLS will not yield consistent estimates when one of the following is true:

- The variance is unbounded.
- There is too much serial dependence in the errors (the dependence does not fade with time).
- A combination of serial dependence and lag dependent variables.

The first two cases are unusual but the third is much more common. Whether iterated feasible GLS avoids consistency problems with the last case is an unanswered research question. For more information see (Davidson and MacKinnon 1993) pages 298-301 or (Gallant 1987) pages 124-125 and (Amemiya 1985) pages 202-203.

One limitation is that parameters can not be shared between the mean equation and the variance equation. This implies that certain GARCH models, cross equation restrictions of parameters, or testing of combinations of parameters in the mean and variance component are not allowed.

Generalized Method of Moments

In GMM, normally the first moment of the mean function is used in the objective function.

$$\begin{aligned} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \epsilon_t \\ \mathbf{E}(\epsilon_t) &= 0 \end{aligned}$$

To add the second moment conditions to the estimation, add the equation

$$\mathbf{E}(\varepsilon_t * \varepsilon_t - h_t) = 0$$

to the model. For example if you wanted to estimate σ for linear example above, you can write

```
proc model data=s;
   y = a * x + b;
   eq.two = resid.y**2 - sigma**2;
fit y two/ gmm;
instruments x;
run;
```

This is a popular way to estimate a continuous-time interest rate processes (see (Chan, et al 1992)). The H.var syntax will automatically generate this system of equations.

To further take advantage of the information obtained about the variance, the moment equations can be modified to

$$\begin{aligned}\mathbf{E}(\varepsilon_t/\sqrt{h_t}) &= 0 \\ \mathbf{E}(\varepsilon_t * \varepsilon_t - h_t) &= 0\end{aligned}$$

For the above example, this can be written as

```
proc model data=s;
   y = a * x + b;
   eq.two = resid.y**2 - sigma**2;
   resid.y = resid.y / sigma;
fit y two/ gmm;
instruments x;
run;
```

Note that, if the error model is misspecified in this form of the GMM model, the parameter estimates may be inconsistent.

Full Information Maximum Likelihood

For FIML estimation of variance functions, the concentrated likelihood below is used as the objective function. That is, the mean function will be coupled with the variance function and the system will be solved simultaneously.

$$\begin{aligned}l_n(\phi) &= \frac{ng}{2}(1+\ln(2\pi)) - \sum_{t=1}^{n} \ln\left(\left|\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t}\right|\right) \\ &\quad + \frac{1}{2}\sum_{t=1}^{n}\sum_{i=1}^{g}\left(\ln(h_{t,i}) + \mathbf{q}_i(\mathbf{y}_t, \mathbf{x}_t, \theta)^2/h_{t,i}\right)\end{aligned}$$

where g is the number of equations in the system.

The HESSIAN=GLS option is not available for FIML estimation involving variance functions. The matrix used when HESSIAN=CROSS is specified is a cross products matrix which has been enhanced by the dual quasi-newton approximation.

Examples

You can specify a GARCH(1,1) model as follows:

```
proc model data=modloc.usd_jpy;

             /* Mean model --------*/
   jpyret = intercept ;

             /* Variance model ----------------*/
   h.jpyret = arch0 + arch1 * xlag( resid.jpyret ** 2, mse.jpyret  )
              + garch1 * xlag(h.jpyret, mse.jpyret) ;

   bounds arch0 arch1 garch1 >= 0;

fit jpyret/method=marquardt fiml;
run;
```

Note that the BOUNDS statement was used to ensure that the parameters were positive, a requirement for GARCH models.

EGARCH models are used because there is no restrictions on the parameters. You can specify a EGARCH(1,1) model as follows:

```
proc model data=sasuser.usd_dem ;

        /* Mean model ----------*/
  demret = intercept ;

           /* Variance model ----------------*/
  if ( _OBS_ =1 )  then
    h.demret = exp( earch0/ (1. - egarch1)  );
  else
    h.demret = exp( earch0 + earch1 * zlag( g)
                       + egarch1 * log(zlag(h.demret)));
  g = theta * nresid.demret + abs( nresid.demret ) - sqrt(2/3.1415);

                     /* Fit and save the model */
fit demret/method=marquardt fiml  maxiter=100
run;
```

Ordinary Differential Equations

Ordinary differential equations (ODEs) are also called *initial value problems* because a time zero value for each first-order differential equation is needed. The following is an example of a first-order system of ODEs:

$$y' = -0.1y + 2.5z^2$$

$$\begin{aligned} z' &= -z \\ y_0 &= 0 \\ z_0 &= 1 \end{aligned}$$

Note that you must provide an initial value for each ODE.

As a reminder, any *n*-order differential equation can be modeled as a system of first-order differential equations. For example, consider the differential equation

$$\begin{aligned} y^{''} &= by^{'} + cy \\ y_0 &= 0 \\ y_0^{'} &= 1 \end{aligned}$$

which can be written as the system of differential equations

$$\begin{aligned} y^{'} &= z \\ z^{'} &= by^{'} + cy \\ y_0 &= 0 \\ z_0 &= 1 \end{aligned}$$

This differential system can be simulated as follows:

```
data t;
   time=0; output;
   time=1; output;
   time=2; output;
run;

proc model data=t ;
   dependent y 0 z 1;
   parm b -2 c -4;
      /* Solve  y’’=b y’ + c y --------------*/

   dert.y = z;
   dert.z = b * dert.y + c * y;

   solve y z / dynamic solveprint;
run;
```

The preceding statements produce the following output. These statements produce additional output, which is not shown.

```
                           The MODEL Procedure
                         Simultaneous Simulation

              Observation    1    Missing       2    CC    -1.000000
                                  Iterations    0


                                  Solution Values

                                     y              z

                              0.000000       1.000000

Observation    2    Iterations    0    CC     0.000000    ERROR.y     0.000000


                                  Solution Values

                                     y              z

                             0.2096398      -.2687053

Observation    3    Iterations    0    CC     9.464802    ERROR.y    -0.234405


                                  Solution Values

                                     y              z

                             -.0247649      -.1035929
```

The differential variables are distinguished by the derivative with respect to time (DERT.) prefix. Once you define the DERT. variable, you can use it on the right-hand side of another equation. The differential equations must be expressed in normal form; implicit differential equations are not allowed, and other terms on the left-hand side are not allowed.

The TIME variable is the *implied with respect to* variable for all DERT. variables. The TIME variable is also the only variable that must be in the input data set.

You can provide initial values for the differential equations in the data set, in the declaration statement (as in the previous example), or in statements in the code. Using the previous example, you can specify the initial values as

```
proc model data=t ;
   dependent y z ;
   parm b -2 c -4;
      /* Solve  y’’=b y’ + c y -------------*/
   if ( time=0 ) then
      do;
         y=0;
         z=1;
      end;
   else
```

```
            do;
               dert.y = z;
               dert.z = b * dert.y + c * y;
            end;
         end;
         solve y z / dynamic solveprint;
      run;
```

If you do not provide an initial value, 0 is used.

DYNAMIC and STATIC Simulation

Note that, in the previous example, the DYNAMIC option was specified in the SOLVE statement. The DYNAMIC and STATIC options work the same for differential equations as they do for dynamic systems. In the differential equation case, the DYNAMIC option makes the initial value needed at each observation the computed value from the previous iteration. For a static simulation, the data set must contain values for the integrated variables. For example, if DERT.Y and DERT.Z are the differential variables, you must include Y and Z in the input data set in order to do a static simulation of the model.

If the simulation is dynamic, the initial values for the differential equations are obtained from the data set, if they are available. If the variable is not in the data set, you can specify the initial value in a declaration statement. If you do not specify an initial value, the value of 0.0 is used.

A dynamic solution is obtained by solving one initial value problem for all the data. A graph of a simple dynamic simulation is shown in Figure 20.38. If the time variable for the current observation is less than the time variable for the previous observation, the integration is restarted from this point. This allows for multiple samples in one data file.

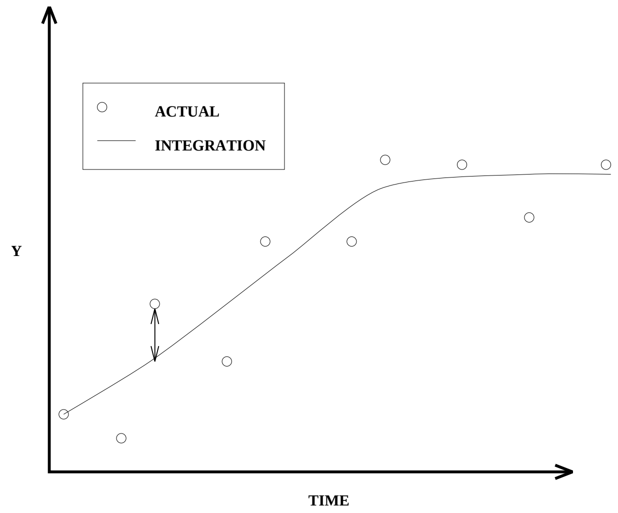


Figure 20.38. Dynamic Solution

In a static solution, n-1 initial value problems are solved using the first n-1 data values as initial values. The equations are integrated using the ith data value as an initial value to the i+1 data value. Figure 20.39 displays a static simulation of noisy data from a simple differential equation. The static solution does not propagate errors in initial values as the dynamic solution does.

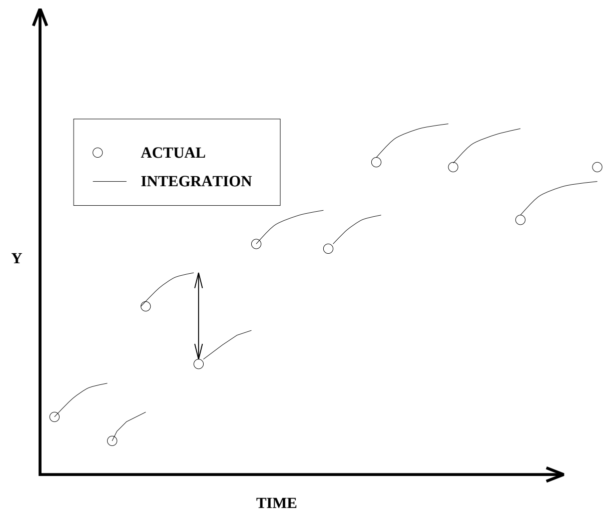


Figure 20.39. Static Solution

For estimation, the DYNAMIC and STATIC options in the FIT statement perform the same functions as they do in the SOLVE statement. Components of differential systems that have missing values or are not in the data set are simulated dynamically. For example, often in multiple compartment kinetic models, only one compartment is monitored. The differential equations describing the unmonitored compartments are simulated dynamically.

For estimation, it is important to have accurate initial values for ODEs that are not in the data set. If an accurate initial value is not known, the initial value can be made an unknown parameter and estimated. This allows for errors in the initial values but increases the number of parameters to estimate by the number of equations.

Estimation of Differential Equations

Consider the kinetic model for the accumulation of mercury (Hg) in mosquito fish (Matis, Miller, and Allen 1991, p. 177). The model for this process is the one-compartment constant infusion model shown in Figure 20.40.

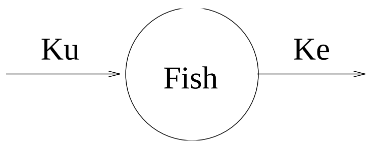


Figure 20.40. One-Compartment Constant Infusion Model

The differential equation that models this process is

$$\begin{aligned} \frac{dconc}{dt} &= k_u - k_e conc \\ conc_0 &= 0 \end{aligned}$$

The analytical solution to the model is

$$conc = (k_u/k_e)(1 - \exp(-\mathrm{k_e t}))$$

The data for the model are

```
data fish;
   input day conc;
   datalines;
0.0    0.0
1.0    0.15
2.0    0.2
3.0    0.26
4.0    0.32
6.0    0.33
;
run;
```

To fit this model in differential form, use the following statements:

```
proc model data=fish;
   parm ku ke;

   dert.conc = ku - ke * conc;

   fit conc / time=day;
run;
```

The results from this estimation are shown in Figure 20.41.

```
                              The MODEL Procedure

                        Nonlinear OLS Parameter Estimates

                                        Approx                      Approx
          Parameter       Estimate     Std Err     t Value       Pr > |t|

          ku              0.180159      0.0312        5.78         0.0044
          ke              0.524661      0.1181        4.44         0.0113
```

Figure 20.41. Static Estimation Results for Fish Model

To perform a dynamic estimation of the differential equation, add the DYNAMIC option to the FIT statement.

```
proc model data=fish;
   parm ku .3 ke .3;

   dert.conc = ku - ke * conc;

   fit conc / time = day dynamic;
run;
```

The equation DERT.CONC is integrated from $conc(0) = 0$. The results from this estimation are shown in Figure 20.42.

```
                          The MODEL Procedure

                   Nonlinear OLS Parameter Estimates

                                      Approx                  Approx
         Parameter       Estimate    Std Err    t Value     Pr > |t|

         ku              0.167109     0.0170       9.84       0.0006
         ke              0.469033     0.0731       6.42       0.0030
```

Figure 20.42. Dynamic Estimation Results for Fish Model

To perform a dynamic estimation of the differential equation and estimate the initial value, use the following statements:

```
proc model data=fish;
   parm ku .3 ke .3 conc0 0;

   dert.conc = ku - ke * conc;

   fit conc initial=(conc = conc0) / time = day dynamic;
run;
```

The INITIAL= option in the FIT statement is used to associate the initial value of a differential equation with a parameter. The results from this estimation are shown in Figure 20.43.

```
                          The MODEL Procedure

                   Nonlinear OLS Parameter Estimates

                                      Approx                  Approx
         Parameter       Estimate    Std Err    t Value     Pr > |t|

         ku              0.164408     0.0230       7.14       0.0057
         ke               0.45949     0.0943       4.87       0.0165
         conc0           0.003798     0.0174       0.22       0.8414
```

Figure 20.43. Dynamic Estimation with Initial Value for Fish Model

Finally, to estimate the fish model using the analytical solution, use the following statements:

```
proc model data=fish;
   parm ku .3  ke .3;

   conc = (ku/ ke)*( 1 -exp(-ke * day));

   fit conc;
run;
```

The results from this estimation are shown in Figure 20.44.

```
                              The MODEL Procedure

                      Nonlinear OLS Parameter Estimates

                                        Approx                      Approx
          Parameter       Estimate     Std Err     t Value     Pr > |t|

          ku              0.167109      0.0170        9.84       0.0006
          ke              0.469033      0.0731        6.42       0.0030
```

Figure 20.44. Analytical Estimation Results for Fish Model

A comparison of the results among the four estimations reveals that the two dynamic estimations and the analytical estimation give nearly identical results (identical to the default precision). The two dynamic estimations are identical because the estimated initial value (0.00013071) is very close to the initial value used in the first dynamic estimation (0). Note also that the static model did not require an initial guess for the parameter values. Static estimation, in general, is more forgiving of bad initial values.

The form of the estimation that is preferred depends mostly on the model and data. If a very accurate initial value is known, then a dynamic estimation makes sense. If, additionally, the model can be written analytically, then the analytical estimation is computationally simpler. If only an approximate initial value is known and not modeled as an unknown parameter, the static estimation is less sensitive to errors in the initial value.

The form of the error in the model is also an important factor in choosing the form of the estimation. If the error term is additive and independent of previous error, then the dynamic mode is appropriate. If, on the other hand, the errors are cumulative, a static estimation is more appropriate. See the section "Monte Carlo Simulation" for an example.

Auxiliary Equations

Auxiliary equations can be used with differential equations. These are equations that need to be satisfied with the differential equations at each point between each data value. They are automatically added to the system, so you do not need to specify them in the SOLVE or FIT statement.

Consider the following example.

The Michaelis-Menten Equations describe the kinetics of an enzyme-catalyzed reaction. The enzyme is E, and S is called the *substrate*. The enzyme first reacts with the substrate to form the enzyme-substrate complex ES, which then breaks down in a second step to form enzyme and products P.

The reaction rates are described by the following system of differential equations:

$$\frac{d[ES]}{dt} = k_1([E]-[ES])[S]-k_2[ES]-k_3[ES]$$

$$\begin{aligned}\frac{d[S]}{dt} &= -k_1([E]-[ES])[S]+k_2[ES] \\ [E] &= [E]_{tot}-[ES]\end{aligned}$$

The first equation describes the rate of formation of ES from E + S. The rate of formation of ES from E + P is very small and can be ignored. The enzyme is in either the complexed or the uncomplexed form. So if the total ($[E]_{tot}$) concentration of enzyme and the amount bound to the substrate is known, $[E]$ can be obtained by conservation.

In this example, the conservation equation is an auxiliary equation and is coupled with the differential equations for integration.

Time Variable

You must provide a time variable in the data set. The name of the time variable defaults to TIME. You can use other variables as the time variable by specifying the TIME= option in the FIT or SOLVE statement. The time intervals need not be evenly spaced. If the time variable for the current observation is less than the time variable for the previous observation, the integration is restarted.

Differential Equations and Goal Seeking

Consider the following differential equation

$$y^{'} = a*x$$

and the data set

```
data t2;
   y=0; time=0; output;
   y=2; time=1; output;
   y=3; time=2; output;
run;
```

The problem is to find values for X that satisfy the differential equation and the data in the data set. Problems of this kind are sometimes referred to as *goal seeking problems* because they require you to search for values of X that will satisfy the goal of Y.

This problem is solved with the following statements:

```
proc model data=t2 ;
   dependent x 0;
   independent y;
   parm a 5;
   dert.y = a * x;
   solve x / out=foo;
run;

proc print data=foo; run;
```

The output from the PROC PRINT statement is shown in Figure 20.45.

```
    Obs     _TYPE_       _MODE_      _ERRORS_         x           y        time

     1      PREDICT     SIMULATE         0        0.00000     0.00000       0
     2      PREDICT     SIMULATE         0        0.80000     2.00000       1
     3      PREDICT     SIMULATE         0       -0.40000     3.00000       2
```

Figure 20.45. Dynamic Solution

Note that an initial value of 0 is provided for the X variable because it is undetermined at TIME = 0.

In the preceding goal seeking example, X is treated as a linear function between each set of data points (see Figure 20.46).

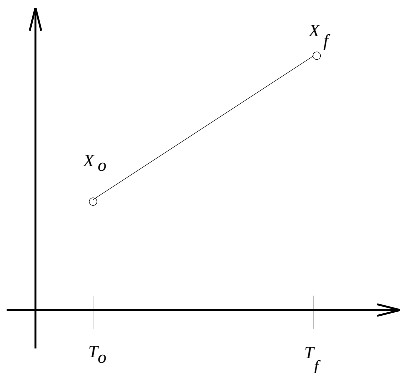


Figure 20.46. Form of X Used for Integration in Goal Seeking

If you integrate $y^{'} = ax$ manually, you have

$$\begin{aligned} x(t) &= \frac{t_f - t}{t_f - t_0} x_0 + \frac{-T_0}{t_f - t_0} x_f \\ y &= \int_{t_o}^{t_f} a x(t) dt \\ &= a \frac{1}{t_f - t_0} (t(t_f x_0 - t_0 x_f) + \frac{1}{2} t^2 (x_f - x_0))|_{t_0}^{t_f} \end{aligned}$$

For observation 2, this reduces to

$$y = \frac{1}{2} a * x_f$$

$$2 \quad = \quad 2.5{*}x_f$$

So $x = 0.8$ for this observation.

Goal seeking for the TIME variable is not allowed.

Restrictions and Bounds on Parameters

Using the BOUNDS and RESTRICT statements, PROC MODEL can compute optimal estimates subject to equality or inequality constraints on the parameter estimates.

Equality restrictions can be written as a vector function

$$\mathbf{h}(\theta) = 0$$

Inequality restrictions are either active or inactive. When an inequality restriction is active, it is treated as an equality restriction. All inactive inequality restrictions can be written as a vector function

$$F(\theta) \geq 0$$

Strict inequalities, such as $(f(\theta) > 0)$, are transformed into inequalities as $f(\theta) \times (1 - \epsilon) - \epsilon \geq 0$, where the tolerance ϵ is controlled by the EPSILON= option on the FIT statement and defaults to 10^{-8}. The *i*th inequality restriction becomes active if $F_i < 0$ and remains active until its Lagrange multiplier becomes negative. Lagrange multipliers are computed for all the nonredundant equality restrictions and all the active inequality restrictions.

For the following, assume the vector $\mathbf{h}(\theta)$ contains all the current active restrictions. The constraint matrix A is

$$A(\hat{\theta}) = \frac{\partial \mathbf{h}(\hat{\theta})}{\partial \hat{\theta}}$$

The covariance matrix for the restricted parameter estimates is computed as

$$Z(Z'HZ)^{-1}Z'$$

where H is Hessian or approximation to the Hessian of the objective function $((X'(\mathrm{diag}(\mathrm{S})^{-1}{\otimes}\mathrm{I})\mathrm{X})$ for OLS), and Z is the last $(np - nc)$ columns of Q. Q is from an LQ factorization of the constraint matrix, *nc* is the number of active constraints, and *np* is the number of parameters. Refer to Gill, Murray, and Wright (1981) for more details on LQ factorization. The covariance column in Table 20.1 summarizes the Hessian approximation used for each estimation method.

The covariance matrix for the Lagrange multipliers is computed as

$$(AH^{-1}A')^{-1}$$

The *p*-value reported for a restriction is computed from a beta distribution rather than a *t*-distribution because the numerator and the denominator of the *t*-ratio for an estimated Lagrange multiplier are not independent.

The Lagrange multipliers for the active restrictions are printed with the parameter estimates. The Lagrange multiplier estimates are computed using the relationship

$$A^{'}\lambda = \mathrm{g}$$

where the dimension of the constraint matrix *A* is the number of constraints by the number of parameters, λ is the vector of Lagrange multipliers, and *g* is the gradient of the objective function at the final estimates.

The final gradient includes the effects of the estimated S matrix. For example, for OLS the final gradient would be:

$$\mathrm{g} = X'(\mathrm{diag(S)}^{-1} \otimes \mathrm{I})\mathrm{r}$$

where *r* is the residual vector. Note that when nonlinear restrictions are imposed, the convergence measure R may have values greater than one for some iterations.

Tests on Parameters

In general, the hypothesis tested can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is a vector valued function of the parameters θ given by the *r* expressions specified on the TEST statement.

Let $\hat{V}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

Let *r* be the dimension of $h(\theta)$ and *n* be the number of observations. Using this notation, the test statistics for the three kinds of tests are computed as follows.

The Wald test statistic is defined as

$$W = h^{'}(\hat{\theta}) \left(A(\hat{\theta})\hat{V}A^{'}(\hat{\theta}) \right)^{-1} h(\hat{\theta})$$

The Wald test is not invariant to reparameterization of the model (Gregory 1985, Gallant 1987, p. 219). For more information on the theoretical properties of the Wald test see Phillips and Park 1988.

The Lagrange multiplier test statistic is

$$R = \lambda' A(\tilde{\theta}) \tilde{V}^{-1} A'(\tilde{\theta}) \lambda$$

where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

The Lagrange multiplier test statistic is equivalent to Rao's efficient score test statistic:

$$R = (\partial L(\tilde{\theta}) / \partial \theta)' \tilde{V}^{-1} (\partial L(\tilde{\theta}) / \partial \theta)$$

where L is the log likelihood function for the estimation method used. For SUR, 3SLS, GMM, and iterated versions of these methods, the likelihood function is computed as

$$L = Objective \times Nobs/2$$

For OLS and 2SLS the Lagrange multiplier test statistic is computed as:

$$R = [(\partial \hat{S}(\tilde{\theta}) / \partial \theta)' \tilde{V}^{-1} (\partial \hat{S}(\tilde{\theta}) / \partial \theta)] / \hat{S}(\tilde{\theta})$$

where $\hat{S}(\tilde{\theta})$ is the corresponding objective function value at the constrained estimate.

The likelihood ratio test statistic is

$$T = 2 \left(L(\tilde{\theta}) - L(\hat{\theta}) \right)$$

where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log likelihood value.

For OLS and 2SLS, the likelihood ratio test statistic is computed as:

$$T = (n - nparms) \times (\hat{S}(\tilde{\theta}) - \hat{S}(\hat{\theta})) / \hat{S}(\hat{\theta})$$

where df is the difference in degrees of freedom for the full and restricted models, and $nparms$ is the number of parameters in the full system.

The Likelihood ratio test is not appropriate for models with nonstationary serially correlated errors (Gallant 1987, p. 139). The likelihood ratio test should not be used for dynamic systems, for systems with lagged dependent variables, or with the FIML estimation method unless certain conditions are met (see Gallant 1987, p. 479).

For each kind of test, under the null hypothesis the test statistic is asymptotically distributed as a χ^2 random variable with *r* degrees of freedom, where *r* is the number of expressions on the TEST statement. The *p*-values reported for the tests are computed from the $\chi^2(r)$ distribution and are only asymptotically valid.

Monte Carlo simulations suggest that the asymptotic distribution of the Wald test is a poorer approximation to its small sample distribution than the other two tests. However, the Wald test has the least computational cost, since it does not require computation of the constrained estimate $\tilde{\theta}$.

The following is an example of using the TEST statement to perform a likelihood ratio test for a compound hypothesis.

```
   test a*exp(-k) = 1-k, d = 0 ,/ lr;
```

It is important to keep in mind that although individual *t* tests for each parameter are printed by default into the parameter estimates table, they are only asymptotically valid for nonlinear models. You should be cautious in drawing any inferences from these *t* tests for small samples.

Hausman Specification Test

Hausman's specification test, or *m*-statistic, can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu (1973). Hausman's *m*-statistic is as follows.

Given two estimators, $\hat{\beta}_0$ and $\hat{\beta}_1$, where under the null hypothesis both estimators are consistent but only $\hat{\beta}_0$ is asymptotically efficient and under the alternative hypothesis only $\hat{\beta}_1$ is consistent, the *m*-statistic is

$$m = \hat{q}'(\hat{V}_1 - \hat{V}_0)^{-}\hat{q}$$

where $\hat{V}_1$ and $\hat{V}_0$ represent consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_1$ and $\hat{\beta}_0$, and

$$q = \hat{\beta}_1 - \hat{\beta}_0$$

The *m*-statistic is then distributed χ^2 with k degrees of freedom, where k is the rank of the matrix $(\hat{V}_1 - \hat{V}_0)$. A generalized inverse is used, as recommended by Hausman (1982).

In the MODEL procedure, Hausman's *m*-statistic can be used to determine if it is necessary to use an instrumental variables method rather than a more efficient OLS estimation. Hausman's *m*-statistic can also be used to compare 2SLS with 3SLS for a class of estimators for which 3SLS is asymptotically efficient (similarly for OLS and SUR).

Hausman's *m*-statistic can also be used, in principle, to test the null hypothesis of normality when comparing 3SLS to FIML. Because of the poor performance of this form of the test, it is not offered in the MODEL procedure. Refer to R.C. Fair (1984, pp. 246-247) for a discussion of why Hausman's test fails for common econometric models.

To perform a Hausman's specification test, specify the HAUSMAN option in the FIT statement. The selected estimation methods are compared using Hausman's *m*-statistic.

In the following example, OLS, SUR, 2SLS, 3SLS, and FIML are used to estimate a model, and Hausman's test is requested.

```
proc model data=one out=fiml2;
   endogenous y1 y2;

   y1 = py2 * y2 + px1 * x1 + interc;
   y2 = py1* y1 + pz1 * z1 + d2;

   fit y1 y2 / ols sur 2sls 3sls fiml hausman;
   instruments x1 z1;
run;
```

The output specified by the HAUSMAN option produces the following results.

```
                         The MODEL Procedure

                  Hausman's Specification Test Results
     Comparing    To            DF    Statistic    Pr > ChiSq

     SUR          OLS            6        17.77        0.0068
     OLS          2SLS           6        13.86        0.0313
     3SLS         OLS            6         0.27        0.9996
     3SLS         2SLS           6        -0.00         .
```

Figure 20.47. Hausman's Specification Test Results

Figure 20.47 indicates that 2SLS, a system estimation method, is preferred over OLS. The model needs an IV estimator but not a full error covariance matrix. Note that the FIML estimation results are not compared.

Chow Tests

The Chow test is used to test for break points or structural changes in a model. The problem is posed as a partitioning of the data into two parts of size n_1 and n_2. The null hypothesis to be tested is

$$H_o : \quad \beta_1 = \beta_2 = \beta$$

where β_1 is estimated using the first part of the data and β_2 is estimated using the second part.

The test is performed as follows (refer to Davidson and MacKinnon 1993, p. 380).

1. The p parameters of the model are estimated.
2. A second linear regression is performed on the residuals, $\hat{u}$, from the nonlinear estimation in step one.

   $$\hat{u} = \hat{X}b + \text{residuals}$$

   where $\hat{X}$ is Jacobian columns that are evaluated at the parameter estimates. If the estimation is an instrumental variables estimation with matrix of instruments W, then the following regression is performed:

   $$\hat{u} = P_{W^*}\hat{X}b + \text{residuals}$$

   where P_{W^*} is the projection matrix.
3. The restricted SSE (RSSE) from this regression is obtained. An SSE for each subsample is then obtained using the same linear regression.
4. The *F* statistic is then

   $$f = \frac{(RSSE - SSE_1 - SSE_2)/p}{(SSE_1 + SSE_2)/(n-2p)}$$

   This test has p and $n - 2p$ degrees of freedom.

Chow's test is not applicable if $\min(n_1, n_2) < p$, since one of the two subsamples does not contain enough data to estimate β. In this instance, the *predictive Chow test* can be used. The predictive Chow test is defined as

$$f = \frac{(RSSE - SSE_1) \times (n_1 - p)}{SSE_1 * n_2}$$

where $n_1 > p$. This test can be derived from the Chow test by noting that the $SSE_2 = 0$ when $n_2 <= p$ and by adjusting the degrees of freedom appropriately.

You can select the Chow test and the predictive Chow test by specifying the CHOW=*arg* and the PCHOW=*arg* options in the FIT statement, where *arg* is either the number of observations in the first sample or a parenthesized list of first sample sizes. If the size of the one of the two groups in which the sample is partitioned is less than the number of parameters, then a Predictive Chow test is automatically used. These tests statistics are not produced for GMM and FIML estimations.

The following is an example of the use of the Chow test.

```
data exp;
   x=0;
   do time=1 to 100;
      if time=50 then x=1;
      y = 35 * exp( 0.01 * time ) + rannor( 123 ) + x * 5;
      output;
   end;
run;
```

```
proc model data=exp;
   parm zo 35 b;
      dert.z = b * z;
      y=z;
   fit y init=(z=zo) / chow =(40 50 60) pchow=90;
run;
```

The data set introduced an artificial structural change into the model (the structural change effects the intercept parameter). The output from the requested Chow tests are shown in Figure 20.48.

```
                              The MODEL Procedure

                             Structural Change Test

                             Break
     Test                    Point     Num DF     Den DF     F Value      Pr > F

     Chow                       40          2         96       12.95      <.0001
     Chow                       50          2         96      101.37      <.0001
     Chow                       60          2         96       26.43      <.0001
     Predictive Chow            90         11         87        1.86      0.0566
```

Figure 20.48. Chow's Test Results

Profile Likelihood Confidence Intervals

Wald-based and likelihood ratio-based confidence intervals are available in the MODEL procedure for computing a confidence interval on an estimated parameter. A confidence interval on a parameter θ can be constructed by inverting a Wald-based or a likelihood ratio-based test.

The approximate $100(1-\alpha)$ % Wald confidence interval for a parameter θ is

$$\hat{\theta} \pm z_{1-\alpha/2}\hat{\sigma}$$

where z_p is the $100p$th percentile of the standard normal distribution, $\hat{\theta}$ is the maximum likelihood estimate of θ, and $\hat{\sigma}$ is the standard error estimate of $\hat{\theta}$.

A likelihood ratio-based confidence interval is derived from the χ^2 distribution of the generalized likelihood ratio test. The approximate $1-\alpha$ confidence interval for a parameter θ is

$$\theta : 2[l(\hat{\theta}) - l(\theta)] \leq q_{1,1-\alpha} = 2l^*$$

where $q_{1,1-\alpha}$ is the $(1-\alpha)$ quantile of the χ^2 with one degree of freedom, and $l(\theta)$ is the log likelihood as a function of one parameter. The endpoints of a confidence interval are the zeros of the function $l(\theta) - l^*$. Computing a likelihood ratio-based confidence interval is an iterative process. This process must be performed twice for each parameter, so the computational cost is considerable. Using a modified form of the algorithm recommended by Venzon and Moolgavkar (1988), you can determine that the cost of each endpoint computation is approximately the cost of estimating the original system.

To request confidence intervals on estimated parameters, specify the following option in the FIT statement:

PRL= WALD | LR | BOTH

By default the PRL option produces 95% likelihood ratio confidence limits. The coverage of the confidence interval is controlled by the ALPHA= option in the FIT statement.

The following is an example of the use of the confidence interval options.

```
data exp;
   do time = 1 to 20;
      y = 35 * exp( 0.01 * time ) + 5*rannor( 123 );
   output;
   end;
run;

proc model data=exp;
   parm zo 35 b;
      dert.z = b * z;
      y=z;
   fit y init=(z=zo) / prl=both;
   test zo = 40.475437 ,/lr;
run;
```

The output from the requested confidence intervals and the TEST statement are shown in Figure 20.49

```
                           The MODEL Procedure

                   Nonlinear OLS Parameter Estimates

                                    Approx                      Approx
          Parameter       Estimate     Std Err     t Value      Pr > |t|

          zo              36.58933      1.9471       18.79       <.0001
          b               0.006497     0.00464        1.40       0.1780


                              Test Results

Test            Type            Statistic    Pr > ChiSq    Label

Test0           L.R.                 3.81        0.0509    zo = 40.475437


                             Parameter Wald
                          95% Confidence Intervals
                Parameter          Value        Lower        Upper

                zo               36.5893      32.7730      40.4056
                b                0.00650     -0.00259       0.0156


                        Parameter Likelihood Ratio
                          95% Confidence Intervals
                Parameter          Value        Lower        Upper

                zo               36.5893      32.8381      40.4921
                b                0.00650     -0.00264       0.0157
```

Figure 20.49. Confidence Interval Estimation

Note that the likelihood ratio test reported the probability that $zo = 40.47543$ is 5% but $zo = 40.47543$ is the upper bound of a 95% confidence interval. To understand this conundrum, note that the TEST statement is using the likelihood ratio statistic to test the null hypothesis $H_0 : zo = 40.47543$ with the alternate that $H_a : zo \neq 40.47543$. The upper confidence interval can be viewed as a test with the null hypothesis $H_0 : zo <= 40.47543$.

Choice of Instruments

Several of the estimation methods supported by PROC MODEL are instrumental variables methods. There is no standard method for choosing instruments for nonlinear regression. Few econometric textbooks discuss the selection of instruments for nonlinear models. Refer to Bowden, R.J. and Turkington, D.A. (1984, p. 180-182) for more information.

The purpose of the instrumental projection is to purge the regressors of their correlation with the residual. For nonlinear systems, the regressors are the partials of the residuals with respect to the parameters.

Possible instrumental variables include

- any variable in the model that is independent of the errors

- lags of variables in the system
- derivatives with respect to the parameters, if the derivatives are independent of the errors
- low degree polynomials in the exogenous variables
- variables from the data set or functions of variables from the data set.

Selected instruments must not

- depend on any variable endogenous with respect to the equations estimated
- depend on any of the parameters estimated
- be lags of endogenous variables if there is serial correlation of the errors.

If the preceding rules are satisfied and there are enough observations to support the number of instruments used, the results should be consistent and the efficiency loss held to a minimum.

You need at least as many instruments as the maximum number of parameters in any equation, or some of the parameters cannot be estimated. Note that *number of instruments* means linearly independent instruments. If you add an instrument that is a linear combination of other instruments, it has no effect and does not increase the effective number of instruments.

You can, however, use too many instruments. In order to get the benefit of instrumental variables, you must have more observations than instruments. Thus, there is a trade-off; the instrumental variables technique completely eliminates the simultaneous equation bias only in large samples. In finite samples, the larger the excess of observations over instruments, the more the bias is reduced. Adding more instruments may improve the efficiency, but after some point efficiency declines as the excess of observations over instruments becomes smaller and the bias grows.

The instruments used in an estimation are printed out at the beginning of the estimation. For example, the following statements produce the instruments list shown in Figure 20.50.

```
proc model data=test2;
   exogenous x1 x2;
   parms b1 a1 a2 b2 2.5 c2 55;
   y1 = a1 * y2 + b1 * exp(x1);
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2;
   fit y1 y2 / n2sls;
   inst b1 b2 c2 x1 ;
run;
```

```
                              The MODEL Procedure

                          The 2 Equations to Estimate

                          y1 =  F(b1, a1(y2))
                          y2 =  F(a2(y1), b2, c2)
                 Instruments  1 x1 @y1/@b1 @y2/@b2 @y2/@c2
```

Figure 20.50. Instruments Used Message

This states that an intercept term, the exogenous variable X1, and the partial derivatives of the equations with respect to B1, B2, and C2, were used as instruments for the estimation.

Examples

Suppose that Y1 and Y2 are endogenous variables, that X1 and X2 are exogenous variables, and that A, B, C, D, E, F, and G are parameters. Consider the following model:

```
   y1 = a + b * x1 + c * y2 + d * lag(y1);
   y2 = e + f * x2 + g * y1;
   fit y1 y2;
   instruments exclude=(c g);
```

The INSTRUMENTS statement produces X1, X2, LAG(Y1), and an intercept as instruments.

In order to estimate the Y1 equation by itself, it is necessary to include X2 explicitly in the instruments since F, in this case, is not included in the estimation

```
   y1 = a + b * x1 + c * y2 + d * lag(y1);
   y2 = e + f * x2 + g * y1;
   fit y1;
   instruments x2 exclude=(c);
```

This produces the same instruments as before. You can list the parameter associated with the lagged variable as an instrument instead of using the EXCLUDE= option. Thus, the following is equivalent to the previous example:

```
   y1 = a + b * x1 + c * y2 + d * lag(y1);
   y2 = e + f * x2 + g * y1;
   fit y1;
   instruments x1 x2 d;
```

For an example of declaring instruments when estimating a model involving identities, consider Klein's Model I

```
proc model data=klien;
   endogenous c p w i x wsum k y;
   exogenous  wp g t year;
   parms c0-c3 i0-i3 w0-w3;
   a: c = c0 + c1 * p + c2 * lag(p) + c3 * wsum;
   b: i = i0 + i1 * p + i2 * lag(p) + i3 * lag(k);
   c: w = w0 + w1 * x + w2 * lag(x) + w3 * year;
   x = c + i + g;
   y = c + i + g-t;
   p = x-w-t;
   k = lag(k) + i;
   wsum = w + wp;
```

The three equations to estimate are identified by the labels A, B, and C. The parameters associated with the predetermined terms are C2, I2, I3, W2, and W3 (and the intercepts, which are automatically added to the instruments). In addition, the system includes five identities that contain the predetermined variables G, T, LAG(K), and WP. Thus, the INSTRUMENTS statement can be written as

```
lagk = lag(k);
instruments c2 i2 i3 w2 w3 g t wp lagk;
```

where LAGK is a program variable used to hold LAG(K). However, this is more complicated than it needs to be. Except for LAG(K), all the predetermined terms in the identities are exogenous variables, and LAG(K) is already included as the coefficient of I3. There are also more parameters for predetermined terms than for endogenous terms, so you might prefer to use the EXCLUDE= option. Thus, you can specify the same instruments list with the simpler statement

```
instruments _exog_ exclude=(c1 c3 i1 w1);
```

To illustrate the use of polynomial terms as instrumental variables, consider the following model:

```
y1 = a + b * exp( c * x1 ) + d * log( x2 ) + e * exp( f * y2 );
```

The parameters are A, B, C, D, E, and F, and the right-hand-side variables are X1, X2, and Y2. Assume that X1 and X2 are exogenous (independent of the error), while Y2 is endogenous. The equation for Y2 is not specified, but assume that it includes the variables X1, X3, and Y1, with X3 exogenous, so the exogenous variables of the full system are X1, X2, and X3. Using as instruments quadratic terms in the exogenous variables, the model is specified to PROC MODEL as follows.

```
proc model;
   parms a b c d e f;
   y1 = a + b * exp( c * x1 ) + d * log( x2 ) + e * exp( f * y2 );
   instruments inst1-inst9;
```

```
      inst1 = x1; inst2 = x2; inst3 = x3;
      inst4 = x1 * x1; inst5 = x1 * x2; inst6 = x1 * x3;
      inst7 = x2 * x2; inst8 = x2 * x3; inst9 = x3 * x3;
      fit y1 / 2sls;
   run;
```

It is not clear what degree polynomial should be used. There is no way to know how good the approximation is for any degree chosen, although the first-stage R^2s may help the assessment.

First-Stage R^2s

When the FSRSQ option is used on the FIT statement, the MODEL procedure prints a column of first-stage R^2 (FSRSQ) statistics along with the parameter estimates. The FSRSQ measures the fraction of the variation of the derivative column associated with the parameter that remains after projection through the instruments.

Ideally, the FSRSQ should be very close to 1.00 for exogenous derivatives. If the FSRSQ is small for an endogenous derivative, it is unclear whether this reflects a poor choice of instruments or a large influence of the errors in the endogenous right-hand-side variables. When the FSRSQ for one or more parameters is small, the standard errors of the parameter estimates are likely to be large.

Note that you can make all the FSRSQs larger (or 1.00) by including more instruments, because of the disadvantage discussed previously. The FSRSQ statistics reported are unadjusted R^2s and do not include a degrees-of-freedom correction.

Autoregressive Moving Average Error Processes

Autoregressive moving average error processes (ARMA errors) and other models involving lags of error terms can be estimated using FIT statements and simulated or forecast using SOLVE statements. ARMA models for the error process are often used for models with autocorrelated residuals. The %AR macro can be used to specify models with autoregressive error processes. The %MA macro can be used to specify models with moving average error processes.

Autoregressive Errors

A model with first-order autoregressive errors, AR(1), has the form

$$y_t = f(x_t, \theta) + \mu_t$$

$$\mu_t = \phi\mu_{t-1} + \epsilon_t$$

while an AR(2) error process has the form

$$\mu_t = \phi_1\mu_{t-1} + \phi_2\mu_{t-2} + \epsilon_t$$

and so forth for higher-order processes. Note that the ϵ_t's are independent and identically distributed and have an expected value of 0.

An example of a model with an AR(2) component is

$$y = \alpha + \beta x_1 + \mu_t$$

$$\mu_t = \phi_1 \mu_{t-1} + \phi_2 \mu_{t-2} + \epsilon_t$$

You would write this model as follows:

```
proc model data=in;
   parms a b p1 p2;
   y = a + b * x1 + p1 * zlag1(y - (a + b * x1)) +
         p2 * zlag2(y - (a + b * x1));
   fit y;
run;
```

or equivalently using the %AR macro as

```
proc model data=in;
   parms a b;
   y = a + b * x1;
   %ar( y, 2 );
   fit y;
run;
```

Moving Average Models

A model with first-order moving average errors, MA(1), has the form

$$y_t = f(x_t) + \mu_t$$

$$\mu_t = \epsilon_t - \theta_1 \epsilon_{t-1}$$

where ϵ_t is identically and independently distributed with mean zero. An MA(2) error process has the form

$$\mu_t = \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2}$$

and so forth for higher-order processes.

For example, you can write a simple linear regression model with MA(2) moving average errors as

```
proc model data=inma2;
   parms a b ma1 ma2;
   y = a + b * x + ma1 * zlag1( resid.y ) +
       ma2 * zlag2( resid.y );
   fit;
run;
```

where MA1 and MA2 are the moving average parameters.

Note that RESID.Y is automatically defined by PROC MODEL as

```
pred.y = a + b * x + ma1 * zlag1( resid.y ) +
       ma2 * zlag2( resid.y );
resid.y = pred.y - actual.y;
```

Note that RESID.Y is ϵ_t.

The ZLAG function must be used for MA models to truncate the recursion of the lags. This ensures that the lagged errors start at zero in the lag-priming phase and do not propagate missing values when lag-priming period variables are missing, and ensures that the future errors are zero rather than missing during simulation or forecasting. For details on the lag functions, see the section "Lag Logic."

This model written using the %MA macro is

```
proc model data=inma2;
   parms a b;
   y =  a + b * x;
   %ma(y, 2);
   fit;
run;
```

General Form for ARMA Models

The general ARMA(*p*,*q*) process has the following form

$$\mu_t = \phi_1 \mu_{t-1} + \ldots + \phi_p \mu_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \ldots - \theta_q \epsilon_{t-q}$$

An ARMA(*p*,*q*) model can be specified as follows

```
yhat =  ... compute structural predicted value here ... ;
yarma = ar1 * zlag1( y - yhat ) + ...  /* ar part */
                                + ar(p) * zlag(p)( y - yhat )
     + ma1 * zlag1( resid.y )   + ...  /* ma part */
                                + ma(q) * zlag(q)( resid.y );
y = yhat + yarma;
```

where AR*i* and MA*j* represent the autoregressive and moving average parameters for the various lags. You can use any names you want for these variables, and there are many equivalent ways that the specification could be written.

Vector ARMA processes can also be estimated with PROC MODEL. For example, a two-variable AR(1) process for the errors of the two endogenous variables Y1 and Y2 can be specified as follows

```
y1hat =  ... compute structural predicted value here ... ;

y1    = y1hat + ar1_1 * zlag1( y1 - y1hat )   /* ar part y1,y1 */
              + ar1_2 * zlag1( y2 - y2hat );  /* ar part y1,y2 */

y21hat =  ... compute structural predicted value here ... ;

y2     = y2hat + ar2_2 * zlag1( y2 - y2hat )   /* ar part y2,y2 */
               + ar2_1 * zlag1( y1 - y1hat );  /* ar part y2,y1 */
```

Convergence Problems with ARMA Models

ARMA models can be difficult to estimate. If the parameter estimates are not within the appropriate range, a moving average model's residual terms will grow exponentially. The calculated residuals for later observations can be very large or can overflow. This can happen either because improper starting values were used or because the iterations moved away from reasonable values.

Care should be used in choosing starting values for ARMA parameters. Starting values of .001 for ARMA parameters usually work if the model fits the data well and the problem is well-conditioned. Note that an MA model can often be approximated by a high order AR model, and vice versa. This may result in high collinearity in mixed ARMA models, which in turn can cause serious ill-conditioning in the calculations and instability of the parameter estimates.

If you have convergence problems while estimating a model with ARMA error processes, try to estimate in steps. First, use a FIT statement to estimate only the structural parameters with the ARMA parameters held to zero (or to reasonable prior estimates if available). Next, use another FIT statement to estimate the ARMA parameters only, using the structural parameter values from the first run. Since the values of the structural parameters are likely to be close to their final estimates, the ARMA parameter estimates may now converge. Finally, use another FIT statement to produce simultaneous estimates of all the parameters. Since the initial values of the parameters are now likely to be quite close to their final joint estimates, the estimates should converge quickly if the model is appropriate for the data.

AR Initial Conditions

The initial lags of the error terms of AR(p) models can be modeled in different ways. The autoregressive error startup methods supported by SAS/ETS procedures are the following:

CLS	conditional least squares (ARIMA and MODEL procedures)
ULS	unconditional least squares (AUTOREG, ARIMA, and MODEL procedures)
ML	maximum likelihood (AUTOREG, ARIMA, and MODEL procedures)
YW	Yule-Walker (AUTOREG procedure only)
HL	Hildreth-Lu, which deletes the first p observations (MODEL procedure only)

See Chapter 12, for an explanation and discussion of the merits of various AR(p) startup methods.

The CLS, ULS, ML, and HL initializations can be performed by PROC MODEL. For AR(1) errors, these initializations can be produced as shown in Table 20.2. These methods are equivalent in large samples.

Table 20.2. Initializations Performed by PROC MODEL: AR(1) ERRORS

Method	**Formula**
conditional least squares	Y=YHAT+AR1*ZLAG1(Y-YHAT);

unconditional least squares	Y=YHAT+AR1*ZLAG1(Y-YHAT);
	IF _OBS_=1 THEN
	RESID.Y=SQRT(1-AR1**2)*RESID.Y;

maximum likelihood	Y=YHAT+AR1*ZLAG1(Y-YHAT);
	W=(1-AR1**2)**(-1/(2*_NUSED_));
	IF _OBS_=1 THEN W=W*SQRT(1-AR1**2);
	RESID.Y=W*RESID.Y;

Hildreth-Lu	Y=YHAT+AR1*LAG1(Y-YHAT);

MA Initial Conditions

The initial lags of the error terms of MA(q) models can also be modeled in different ways. The following moving average error startup paradigms are supported by the ARIMA and MODEL procedures:

ULS	unconditional least squares
CLS	conditional least squares
ML	maximum likelihood

The conditional least-squares method of estimating moving average error terms is not optimal because it ignores the startup problem. This reduces the efficiency of the estimates, although they remain unbiased. The initial lagged residuals, extending before the start of the data, are assumed to be 0, their unconditional expected value. This introduces a difference between these residuals and the generalized least-squares residuals for the moving average covariance, which, unlike the autoregressive model, persists through the data set. Usually this difference converges quickly to 0, but for nearly noninvertible moving average processes the convergence is quite slow. To minimize this problem, you should have plenty of data, and the moving average parameter estimates should be well within the invertible range.

This problem can be corrected at the expense of writing a more complex program. Unconditional least-squares estimates for the MA(1) process can be produced by specifying the model as follows:

```
yhat =  ... compute structural predicted value here ... ;
if _obs_ = 1 then do;
   h = sqrt( 1 + ma1 ** 2 );
   y = yhat;
```

```
      resid.y = ( y - yhat ) / h;
      end;
   else do;
      g = ma1 / zlag1( h );
      h = sqrt( 1 + ma1 ** 2 - g ** 2 );
      y = yhat + g * zlag1( resid.y );
      resid.y = ( ( y - yhat) - g * zlag1( resid.y ) ) / h;
      end;
```

Moving-average errors can be difficult to estimate. You should consider using an AR(*p*) approximation to the moving average process. A moving average process can usually be well-approximated by an autoregressive process if the data have not been smoothed or differenced.

The %AR Macro

The SAS macro %AR generates programming statements for PROC MODEL for autoregressive models. The %AR macro is part of SAS/ETS software and no special options need to be set to use the macro. The autoregressive process can be applied to the structural equation errors or to the endogenous series themselves.

The %AR macro can be used for

- univariate autoregression
- unrestricted vector autoregression
- restricted vector autoregression.

Univariate Autoregression

To model the error term of an equation as an autoregressive process, use the following statement after the equation:

```
   %ar( varname, nlags )
```

For example, suppose that Y is a linear function of X1 and X2, and an AR(2) error. You would write this model as follows:

```
   proc model data=in;
      parms a b c;
      y = a + b * x1 + c * x2;
      %ar( y, 2 )
      fit y / list;
   run;
```

The calls to %AR must come *after* all of the equations that the process applies to.

The preceding macro invocation, %AR(y,2), produces the statements shown in the LIST output in Figure 20.51.

```
                          The MODEL Procedure

                        Listing of Compiled Program Code
          Stmt     Line:Col      Statement as Parsed

             1     5738:50       PRED.y = a + b * x1 + c * x2;
             1     5738:50       RESID.y = PRED.y - ACTUAL.y;
             1     5738:50       ERROR.y = PRED.y - y;
             2     7987:23       _PRED__y = PRED.y;
             3     8003:15       #OLD_PRED.y = PRED.y + y_l1
                                 * ZLAG1( y - _PRED__y ) + y_l2
                                 * ZLAG2( y - _PRED__y );
             3     8003:15       PRED.y = #OLD_PRED.y;
             3     8003:15       RESID.y = PRED.y - ACTUAL.y;
             3     8003:15       ERROR.y = PRED.y - y;
```

Figure 20.51. LIST Option Output for an AR(2) Model

The _PRED__ prefixed variables are temporary program variables used so that the lags of the residuals are the correct residuals and not the ones redefined by this equation. Note that this is equivalent to the statements explicitly written in the "General Form for ARMA Models" earlier in this section.

You can also restrict the autoregressive parameters to zero at selected lags. For example, if you wanted autoregressive parameters at lags 1, 12, and 13, you can use the following statements:

```
proc model data=in;
   parms a b c;
   y = a + b * x1 + c * x2;
   %ar( y, 13, , 1 12 13 )
   fit y / list;
run;
```

These statements generate the output shown in Figure 20.52.

```
                          The MODEL Procedure

                        Listing of Compiled Program Code
          Stmt     Line:Col      Statement as Parsed

             1     8182:50       PRED.y = a + b * x1 + c * x2;
             1     8182:50       RESID.y = PRED.y - ACTUAL.y;
             1     8182:50       ERROR.y = PRED.y - y;
             2     8631:23       _PRED__y = PRED.y;
             3     8647:15       #OLD_PRED.y = PRED.y + y_l1 * ZLAG1( y -
                                 _PRED__y ) + y_l12 * ZLAG12( y -
                                 _PRED__y ) + y_l13 * ZLAG13(
                                 y - _PRED__y );
             3     8647:15       PRED.y = #OLD_PRED.y;
             3     8647:15       RESID.y = PRED.y - ACTUAL.y;
             3     8647:15       ERROR.y = PRED.y - y;
```

Figure 20.52. LIST Option Output for an AR Model with Lags at 1, 12, and 13

There are variations on the conditional least-squares method, depending on whether observations at the start of the series are used to "warm up" the AR process. By default, the %AR conditional least-squares method uses all the observations and assumes zeros for the initial lags of autoregressive terms. By using the M= option, you can request that %AR use the unconditional least-squares (ULS) or maximum-likelihood (ML) method instead. For example,

```
proc model data=in;
   y = a + b * x1 + c * x2;
   %ar( y, 2, m=uls )
   fit y;
run;
```

Discussions of these methods is provided in the "AR Initial Conditions" earlier in this section.

By using the M=CLS*n* option, you can request that the first *n* observations be used to compute estimates of the initial autoregressive lags. In this case, the analysis starts with observation *n*+1. For example:

```
proc model data=in;
   y = a + b * x1 + c * x2;
   %ar( y, 2, m=cls2 )
   fit y;
run;
```

You can use the %AR macro to apply an autoregressive model to the endogenous variable, instead of to the error term, by using the TYPE=V option. For example, if you want to add the five past lags of Y to the equation in the previous example, you could use %AR to generate the parameters and lags using the following statements:

```
proc model data=in;
   parms a b c;
   y = a + b * x1 + c * x2;
   %ar( y, 5, type=v )
   fit y / list;
run;
```

The preceding statements generate the output shown in Figure 20.53.

```
                              The MODEL Procedure

                        Listing of Compiled Program Code
     Stmt    Line:Col      Statement as Parsed

        1    8892:50       PRED.y = a + b * x1 + c * x2;
        1    8892:50       RESID.y = PRED.y - ACTUAL.y;
        1    8892:50       ERROR.y = PRED.y - y;
        2    9301:15       #OLD_PRED.y = PRED.y + y_l1 * ZLAG1( y )
                           + y_l2 * ZLAG2( y ) + y_l3 * ZLAG3( y )
                           + y_l4 * ZLAG4( y ) + y_l5 * ZLAG5( y );
        2    9301:15       PRED.y = #OLD_PRED.y;
        2    9301:15       RESID.y = PRED.y - ACTUAL.y;
        2    9301:15       ERROR.y = PRED.y - y;
```

Figure 20.53. LIST Option Output for an AR model of Y

This model predicts Y as a linear combination of X1, X2, an intercept, and the values of Y in the most recent five periods.

Unrestricted Vector Autoregression

To model the error terms of a set of equations as a vector autoregressive process, use the following form of the %AR macro after the equations:

```
%ar( process_name, nlags, variable_list )
```

The *process_name* value is any name that you supply for %AR to use in making names for the autoregressive parameters. You can use the %AR macro to model several different AR processes for different sets of equations by using different process names for each set. The process name ensures that the variable names used are unique. Use a short *process_name* value for the process if parameter estimates are to be written to an output data set. The %AR macro tries to construct parameter names less than or equal to eight characters, but this is limited by the length of *name*, which is used as a prefix for the AR parameter names.

The *variable_list* value is the list of endogenous variables for the equations.

For example, suppose that errors for equations Y1, Y2, and Y3 are generated by a second-order vector autoregressive process. You can use the following statements:

```
proc model data=in;
   y1 = ... equation for y1 ...;
   y2 = ... equation for y2 ...;
   y3 = ... equation for y3 ...;
   %ar( name, 2, y1 y2 y3 )
   fit y1 y2 y3;
run;
```

which generates the following for Y1 and similar code for Y2 and Y3:

```
y1 = pred.y1 + name1_1_1*zlag1(y1-name_y1) +
     name1_1_2*zlag1(y2-name_y2) +
     name1_1_3*zlag1(y3-name_y3) +
     name2_1_1*zlag2(y1-name_y1) +
     name2_1_2*zlag2(y2-name_y2) +
     name2_1_3*zlag2(y3-name_y3) ;
```

Only the conditional least-squares (M=CLS or M=CLS*n*) method can be used for vector processes.

You can also use the same form with restrictions that the coefficient matrix be 0 at selected lags. For example, the statements

```
proc model data=in;
   y1 = ... equation for y1 ...;
```

```
      y2 = ... equation for y2 ...;
      y3 = ... equation for y3 ...;
      %ar( name, 3, y1 y2 y3, 1 3 )
      fit y1 y2 y3;
```

apply a third-order vector process to the equation errors with all the coefficients at lag 2 restricted to 0 and with the coefficients at lags 1 and 3 unrestricted.

You can model the three series Y1-Y3 as a vector autoregressive process in the variables instead of in the errors by using the TYPE=V option. If you want to model Y1-Y3 as a function of past values of Y1-Y3 and some exogenous variables or constants, you can use %AR to generate the statements for the lag terms. Write an equation for each variable for the nonautoregressive part of the model, and then call %AR with the TYPE=V option. For example,

```
   proc model data=in;
      parms a1-a3 b1-b3;
      y1 = a1 + b1 * x;
      y2 = a2 + b2 * x;
      y3 = a3 + b3 * x;
      %ar( name, 2, y1 y2 y3, type=v )
      fit y1 y2 y3;
   run;
```

The nonautoregressive part of the model can be a function of exogenous variables, or it may be intercept parameters. If there are no exogenous components to the vector autoregression model, including no intercepts, then assign zero to each of the variables. There must be an assignment to each of the variables before %AR is called.

```
   proc model data=in;
      y1=0;
      y2=0;
      y3=0;
      %ar( name, 2, y1 y2 y3, type=v )
      fit y1 y2 y3;
```

This example models the vector Y=(Y1 Y2 Y3)′ as a linear function only of its value in the previous two periods and a white noise error vector. The model has 18=(3 × 3 + 3 × 3) parameters.

Syntax of the %AR Macro

There are two cases of the syntax of the %AR macro. The first has the general form

%AR *(name, nlag [,endolist [***,***laglist]] [***,M=***method] [***,TYPE=***V])*

where

name specifies a prefix for %AR to use in constructing names of variables needed to define the AR process. If the *endolist* is not specified, the

	endogenous list defaults to *name*, which must be the name of the equation to which the AR error process is to be applied. The *name* value cannot exceed 32 characters.
nlag	is the order of the AR process.
endolist	specifies the list of equations to which the AR process is to be applied. If more than one name is given, an unrestricted vector process is created with the structural residuals of all the equations included as regressors in each of the equations. If not specified, *endolist* defaults to *name*.
laglist	specifies the list of lags at which the AR terms are to be added. The coefficients of the terms at lags not listed are set to 0. All of the listed lags must be less than or equal to *nlag*, and there must be no duplicates. If not specified, the *laglist* defaults to all lags 1 through *nlag*.
M=*method*	specifies the estimation method to implement. Valid values of M= are CLS (conditional least-squares estimates), ULS (unconditional least-squares estimates), and ML (maximum-likelihood estimates). M=CLS is the default. Only M=CLS is allowed when more than one equation is specified. The ULS and ML methods are not supported for vector AR models by %AR.
TYPE=V	specifies that the AR process is to be applied to the endogenous variables themselves instead of to the structural residuals of the equations.

Restricted Vector Autoregression

You can control which parameters are included in the process, restricting those parameters that you do not include to 0. First, use %AR with the DEFER option to declare the variable list and define the dimension of the process. Then, use additional %AR calls to generate terms for selected equations with selected variables at selected lags. For example,

```
proc model data=d;
   y1 = ... equation for y1 ...;
   y2 = ... equation for y2 ...;
   y3 = ... equation for y3 ...;
   %ar( name, 2, y1 y2 y3, defer )
   %ar( name, y1, y1 y2 )
   %ar( name, y2 y3, , 1 )
   fit y1 y2 y3;
run;
```

The error equations produced are

```
y1 = pred.y1 + name1_1_1*zlag1(y1-name_y1) +
     name1_1_2*zlag1(y2-name_y2) + name2_1_1*zlag2(y1-name_y1) +
     name2_1_2*zlag2(y2-name_y2) ;
```

```
   y2 = pred.y2 + name1_2_1*zlag1(y1-name_y1) +
        name1_2_2*zlag1(y2-name_y2) + name1_2_3*zlag1(y3-name_y3) ;
   y3 = pred.y3 + name1_3_1*zlag1(y1-name_y1) +
        name1_3_2*zlag1(y2-name_y2) + name1_3_3*zlag1(y3-name_y3) ;
```

This model states that the errors for Y1 depend on the errors of both Y1 and Y2 (but not Y3) at both lags 1 and 2, and that the errors for Y2 and Y3 depend on the previous errors for all three variables, but only at lag 1.

%AR Macro Syntax for Restricted Vector AR

An alternative use of %AR is allowed to impose restrictions on a vector AR process by calling %AR several times to specify different AR terms and lags for different equations.

The first call has the general form

%AR(*name, nlag, endolist, DEFER)*

where

name	specifies a prefix for %AR to use in constructing names of variables needed to define the vector AR process.
nlag	specifies the order of the AR process.
endolist	specifies the list of equations to which the AR process is to be applied.
DEFER	specifies that %AR is not to generate the AR process but is to wait for further information specified in later %AR calls for the same *name* value.

The subsequent calls have the general form

```
   %AR( name, eqlist, varlist, laglist,TYPE= )
```

where

name	is the same as in the first call.
eqlist	specifies the list of equations to which the specifications in this %AR call are to be applied. Only names specified in the *endolist* value of the first call for the *name* value can appear in the list of equations in *eqlist*.
varlist	specifies the list of equations whose lagged structural residuals are to be included as regressors in the equations in *eqlist*. Only names in the *endolist* of the first call for the *name* value can appear in *varlist*. If not specified, *varlist* defaults to *endolist*.

laglist	specifies the list of lags at which the AR terms are to be added. The coefficients of the terms at lags not listed are set to 0. All of the listed lags must be less than or equal to the value of *nlag*, and there must be no duplicates. If not specified, *laglist* defaults to all lags 1 through *nlag*.

The %MA Macro

The SAS macro %MA generates programming statements for PROC MODEL for moving average models. The %MA macro is part of SAS/ETS software and no special options are needed to use the macro. The moving average error process can be applied to the structural equation errors. The syntax of the %MA macro is the same as the %AR macro except there is no TYPE= argument.

When you are using the %MA and %AR macros combined, the %MA macro must follow the %AR macro. The following SAS/IML statements produce an ARMA(1, (1 3)) error process and save it in the data set MADAT2.

```
 /* use IML module to simulate a MA process  */
proc iml;
   phi={1 .2};
   theta={ 1 .3 0 .5};
   y=armasim(phi, theta, 0,.1, 200,32565);
   create madat2 from y[colname='y'];
   append to y;
quit;
```

The following PROC MODEL statements are used to estimate the parameters of this model using maximum likelihood error structure:

```
title1 'Maximum Likelihood ARMA(1, (1 3))';
proc model data=madat2;
   y=0;
   %ar(y,1,,  M=ml)
   %ma(y,3,,1 3,  M=ml)  /* %MA always after %AR */
   fit y;
run;
```

The estimates of the parameters produced by this run are shown in Figure 20.54.

```
                       Maximum Likelihood ARMA(1, (1 3))

                              The MODEL Procedure

                    Nonlinear OLS Summary of Residual Errors

                  DF      DF                                                Adj
  Equation      Model   Error        SSE        MSE   Root MSE  R-Square     R-Sq

  y                 3     197     2.6383     0.0134     0.1157   -0.0067  -0.0169
  RESID.y                 197     1.9957     0.0101     0.1007


                        Nonlinear OLS Parameter Estimates

                             Approx                  Approx
Parameter      Estimate     Std Err    t Value     Pr > |t|    Label

y_l1           -0.10067      0.1187      -0.85       0.3973    AR(y) y lag1
                                                               parameter
y_m1            -0.1934      0.0939      -2.06       0.0408    MA(y) y lag1
                                                               parameter
y_m3           -0.59384      0.0601      -9.88       <.0001    MA(y) y lag3
                                                               parameter
```

Figure 20.54. Estimates from an ARMA(1, (1 3)) Process

Syntax of the %MA Macro

There are two cases of the syntax for the %MA macro. The first has the general form

%MA *(name, nlag [,endolist [,laglist]] [,***M=***method])*

where

name	specifies a prefix for %MA to use in constructing names of variables needed to define the MA process and is the default *endolist*.
nlag	is the order of the MA process.
endolist	specifies the equations to which the MA process is to be applied. If more than one name is given, CLS estimation is used for the vector process.
laglist	specifies the lags at which the MA terms are to be added. All of the listed lags must be less than or equal to *nlag*, and there must be no duplicates. If not specified, the *laglist* defaults to all lags 1 through *nlag*.
M=*method*	specifies the estimation method to implement. Valid values of M= are CLS (conditional least-squares estimates), ULS (unconditional least-squares estimates), and ML (maximum-likelihood estimates). M=CLS is the default. Only M=CLS is allowed when more than one equation is specified on the *endolist*.

%MA Macro Syntax for Restricted Vector Moving Average

An alternative use of %MA is allowed to impose restrictions on a vector MA process by calling %MA several times to specify different MA terms and lags for different equations.

The first call has the general form

%MA(*name, nlag, endolist,* **DEFER** *)*

where

name	specifies a prefix for %MA to use in constructing names of variables needed to define the vector MA process.
nlag	specifies the order of the MA process.
endolist	specifies the list of equations to which the MA process is to be applied.
DEFER	specifies that %MA is not to generate the MA process but is to wait for further information specified in later %MA calls for the same *name* value.

The subsequent calls have the general form

```
%MA( name, eqlist, varlist, laglist )
```

where

name	is the same as in the first call.
eqlist	specifies the list of equations to which the specifications in this %MA call are to be applied.
varlist	specifies the list of equations whose lagged structural residuals are to be included as regressors in the equations in *eqlist*.
laglist	specifies the list of lags at which the MA terms are to be added.

Distributed Lag Models and the %PDL Macro

In the following example, the variable *y* is modeled as a linear function of *x*, the first lag of *x*, the second lag of *x*, and so forth:

$$y_t = a + b_0 x_t + b_1 x_{t-1} + b_2 x_{t-2} + b_3 x_{t-3} + \ldots + b_n x_{t-l}$$

Models of this sort can introduce a great many parameters for the lags, and there may not be enough data to compute accurate independent estimates for them all. Often, the number of parameters is reduced by assuming that the lag coefficients follow some pattern. One common assumption is that the lag coefficients follow a polynomial in the lag length

$$b_i = \sum_{j=0}^{d} \alpha_j (i)^j$$

where *d* is the degree of the polynomial used. Models of this kind are called *Almon lag models*, *polynomial distributed lag models*, or *PDLs* for short. For example, Figure 20.55 shows the lag distribution that can be modeled with a low order polynomial. Endpoint restrictions can be imposed on a PDL to require that the lag coefficients be 0 at the 0th lag, or at the final lag, or at both.

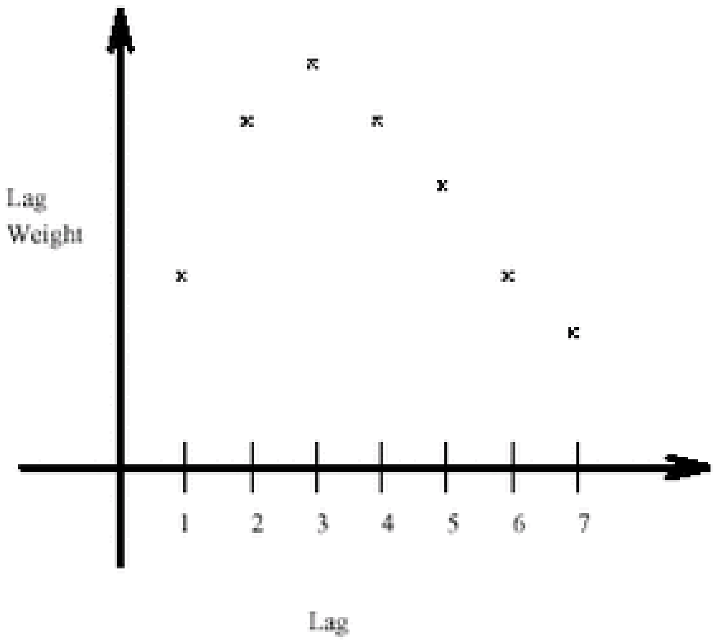


Figure 20.55. Polynomial Distributed Lags

For linear single-equation models, SAS/ETS software includes the PDLREG procedure for estimating PDL models. See Chapter 21, "The PDLREG Procedure," for a more detailed discussion of polynomial distributed lags and an explanation of endpoint restrictions.

Polynomial and other distributed lag models can be estimated and simulated or forecast with PROC MODEL. For polynomial distributed lags, the %PDL macro can generate the needed programming statements automatically.

The %PDL Macro

The SAS macro %PDL generates the programming statements to compute the lag coefficients of polynomial distributed lag models and to apply them to the lags of variables or expressions.

To use the %PDL macro in a model program, you first call it to declare the lag distribution; later, you call it again to apply the PDL to a variable or expression. The first call generates a PARMS statement for the polynomial parameters and assignment statements to compute the lag coefficients. The second call generates an expression that applies the lag coefficients to the lags of the specified variable or expression. A PDL can be declared only once, but it can be used any number of times (that is, the second call can be repeated).

The initial declaratory call has the general form

%PDL *(pdlname, nlags, degree, R=code, OUTEST=dataset)*

where *pdlname* is a name (up to 32 characters) that you give to identify the PDL, *nlags* is the lag length, and *degree* is the degree of the polynomial for the distribution. The R=*code* is optional for endpoint restrictions. The value of *code* can be FIRST (for upper), LAST (for lower), or BOTH (for both upper and lower endpoints). See chapter pdlreg, "The PDLREG Procedure," for a discussion of endpoint restrictions. The option OUTEST=*dataset* creates a data set containing the estimates of the parameters and their covariance matrix.

The later calls to apply the PDL have the general form

```
%PDL( pdlname, expression )
```

where *pdlname* is the name of the PDL and *expression* is the variable or expression to which the PDL is to be applied. The *pdlname* given must be the same as the name used to declare the PDL.

The following statements produce the output in Figure 20.56:

```
proc model data=in list;
   parms int pz;
   %pdl(xpdl,5,2);
   y = int + pz * z + %pdl(xpdl,x);
   %ar(y,2,M=ULS);
   id i;
fit y / out=model1 outresid converge=1e-6;
run;
```

```
                                 The MODEL Procedure

                              Nonlinear OLS  Estimates

                                 Approx                    Approx
Term              Estimate      Std Err    t Value     Pr > |t|    Label

XPDL_L0           1.568788       0.0992      15.81       <.0001    PDL(XPDL,5,2)
                                                                   coefficient for lag0
XPDL_L1           0.564917       0.0348      16.24       <.0001    PDL(XPDL,5,2)
                                                                   coefficient for lag1
XPDL_L2           -0.05063       0.0629      -0.80       0.4442    PDL(XPDL,5,2)
                                                                   coefficient for lag2
XPDL_L3           -0.27785       0.0549      -5.06       0.0010    PDL(XPDL,5,2)
                                                                   coefficient for lag3
XPDL_L4           -0.11675       0.0390      -2.99       0.0173    PDL(XPDL,5,2)
                                                                   coefficient for lag4
XPDL_L5            0.43267       0.1445       2.99       0.0172    PDL(XPDL,5,2)
                                                                   coefficient for lag5
```

Figure 20.56. %PDL Macro ESTIMATE Statement Output

This second example models two variables, Y1 and Y2, and uses two PDLs:

```
proc model data=in;
   parms int1 int2;
   %pdl( logxpdl, 5, 3 )
   %pdl( zpdl, 6, 4 )
   y1 = int1 + %pdl( logxpdl, log(x) ) + %pdl( zpdl, z );
   y2 = int2 + %pdl( zpdl, z );
   fit y1 y2;
run;
```

A (5,3) PDL of the log of X is used in the equation for Y1. A (6,4) PDL of Z is used in the equations for both Y1 and Y2. Since the same ZPDL is used in both equations, the lag coefficients for Z are the same for the Y1 and Y2 equations, and the polynomial parameters for ZPDL are shared by the two equations. See Example 20.5 for a complete example and comparison with PDLREG.

Input Data Sets

DATA= Input Data Set

For FIT tasks, the DATA= option specifies which input data set to use in estimating parameters. Variables in the model program are looked up in the DATA= data set and, if found, their attributes (type, length, label, and format) are set to be the same as those in the DATA= data set (if not defined otherwise within PROC MODEL), and values for the variables in the program are read from the data set.

ESTDATA= Input Data Set

The ESTDATA= option specifies an input data set that contains an observation giving values for some or all of the model parameters. The data set can also contain observations giving the rows of a covariance matrix for the parameters.

Parameter values read from the ESTDATA= data set provide initial starting values for parameters estimated. Observations providing covariance values, if any are present in the ESTDATA= data set, are ignored.

The ESTDATA= data set is usually created by the OUTEST= option in a previous FIT statement. You can also create an ESTDATA= data set with a SAS DATA step program. The data set must contain a numeric variable for each parameter to be given a value or covariance column. The name of the variable in the ESTDATA= data set must match the name of the parameter in the model. Parameters with names longer than 32 characters cannot be set from an ESTDATA= data set. The data set must also contain a character variable _NAME_ of length 32. _NAME_ has a blank value for the observation that gives values to the parameters. _NAME_ contains the name of a parameter for observations defining rows of the covariance matrix.

More than one set of parameter estimates and covariances can be stored in the ESTDATA= data set if the observations for the different estimates are identified by the variable _TYPE_. _TYPE_ must be a character variable of length 8. The TYPE= option is used to select for input the part of the ESTDATA= data set for which the _TYPE_ value matches the value of the TYPE= option.

The following SAS statements generate the ESTDATA= data set shown in Figure 20.57. The second FIT statement uses the TYPE= option to select the estimates from the GMM estimation as starting values for the FIML estimation.

```
         /* Generate test data */
data gmm2;
    do t=1 to 50;
       x1 = sqrt(t) ;
       x2 = rannor(10) * 10;
       y1 = -.002 * x2 * x2 - .05 / x2 - 0.001 * x1 * x1;
       y2 = 0.002* y1 + 2 * x2 * x2 + 50 / x2 + 5 * rannor(1);
       y1 = y1 + 5 * rannor(1);
       z1 = 1; z2 = x1 * x1; z3 = x2 * x2; z4 = 1.0/x2;
       output;
    end;
run;

proc model data=gmm2 ;
   exogenous x1 x2;
   parms a1 a2 b1 2.5 b2 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm kernel=(qs,1,0.2) outest=gmmest;

   fit y1 y2 / fiml type=gmm estdata=gmmest;
run;

proc print data=gmmest;
run;
```

```
                 _
                 S       _
   _  _          T       N
   N  T          A       U
   A  Y          T       S
 O M  P          U       E
 b E  E          S       D      a          a         b        b        c         d
 s _  _          _       _      1          2         1        2        2         1

 1   3SLS 0 Converged 50 -.002229607 -1.25002 0.025827 1.99609 49.8119 -0.44533
 2   GMM  0 Converged 50 -.002013073 -1.53882 0.014908 1.99419 49.8035 -0.64933
```

Figure 20.57. ESTDATA= Data Set

MISSING= PAIRWISE | DELETE

When missing values are encountered for any one of the equations in a system of equations, the default action is to drop that observation for all of the equations. The new MISSING=PAIRWISE option on the FIT statement provides a different method of handling missing values that avoids losing data for nonmissing equations for the observation. This is especially useful for SUR estimation on equations with unequal numbers of observations.

The option MISSING=PAIRWISE specifies that missing values are tracked on an equation-by-equation basis. The MISSING=DELETE option specifies that the entire observation is omitted from the analysis when any equation has a missing predicted or actual value for the equation. The default is MISSING=DELETE.

When you specify the MISSING=PAIRWISE option, the S matrix is computed as

$$S = D(R'R)D$$

where D is a diagonal matrix that depends on the VARDEF= option, the matrix R is $(\mathbf{r}_1, \ldots, \mathbf{r}_g)$, and $\mathbf{r}_i$ is the vector of residuals for the *i*th equation with r_{ij} replaced with zero when r_{ij} is missing.

For MISSING=PAIRWISE, the calculation of the diagonal element $d_{i,i}$ of **D** is based on n_i, the number of nonmissing observations for the *i*th equation, instead of on *n* or, for VARDEF=WGT or WDF, on the sum of the weights for the nonmissing observations for the *i*th equation instead of on the sum of the weights for all observations. Refer to the description of the VARDEF= option for the definition of **D**.

The degrees of freedom correction for a shared parameter is computed using the average number of observations used in its estimation.

The MISSING=PAIRWISE option is not valid for the GMM and FIML estimation methods.

For the instrumental variables estimation methods (2SLS, 3SLS), when an instrument is missing for an observation, that observation is dropped for all equations,

regardless of the MISSING= option.

PARMSDATA= Input Data Set

The option PARMSDATA= reads values for all parameters whose names match the names of variables in the PARMSDATA= data set. Values for any or all of the parameters in the model can be reset using the PARMSDATA= option. The PARMSDATA= option goes on the PROC MODEL statement, and the data set is read before any FIT or SOLVE statements are executed.

Together, the OUTPARMS= and PARMSDATA= options allow you to change part of a model and recompile the new model program without the need to reestimate equations that were not changed.

Suppose you have a large model with parameters estimated and you now want to replace one equation, Y, with a new specification. Although the model program must be recompiled with the new equation, you don't need to reestimate all the equations, just the one that changed.

Using the OUTPARMS= and PARMSDATA= options, you could do the following:

```
proc model model=oldmod outparms=temp; run;
proc model outmodel=newmod parmsdata=temp data=in;
   ...  include new model definition with changed y eq. here ...
   fit y;
run;
```

The model file NEWMOD will then contain the new model and its estimated parameters plus the old models with their original parameter values.

SDATA= Input Data Set

The SDATA= option allows a cross-equation covariance matrix to be input from a data set. The **S** matrix read from the SDATA= data set, specified in the FIT statement, is used to define the objective function for the OLS, N2SLS, SUR, and N3SLS estimation methods and is used as the initial **S** for the methods that iterate the **S** matrix.

Most often, the SDATA= data set has been created by the OUTS= or OUTSUSED= option on a previous FIT statement. The OUTS= and OUTSUSED= data sets from a FIT statement can be read back in by a FIT statement in the same PROC MODEL step.

You can create an input SDATA= data set using the DATA step. PROC MODEL expects to find a character variable _NAME_ in the SDATA= data set as well as variables for the equations in the estimation or solution. For each observation with a _NAME_ value matching the name of an equation, PROC MODEL fills the corresponding row of the **S** matrix with the values of the names of equations found in the data set. If a row or column is omitted from the data set, a 1 is placed on the diagonal for the row or column. Missing values are ignored, and since the **S** matrix is symmetric, you can include only a triangular part of the **S** matrix in the SDATA= data set with the omitted part indicated by missing values. If the SDATA= data set contains multiple observations with the same _NAME_, the last values supplied for the _NAME_ are used. The structure of the expected data set is further described in the "OUTS=Data Set" section.

Use the TYPE= option on the PROC MODEL or FIT statement to specify the type of estimation method used to produce the **S** matrix you want to input.

The following SAS statements are used to generate an **S** matrix from a GMM and a 3SLS estimation and to store that estimate in the data set GMMS:

```
proc model data=gmm2 ;
   exogenous x1 x2;
   parms a1 a2 b1 2.5 b2 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm kernel=(qs,1,0.2) outest=gmmest outs=gmms;
run;
```

The data set GMMS is shown in Figure 20.58.

```
          Obs     _NAME_     _TYPE_     _NUSED_        y1          y2

            1       y1       3SLS          50      27.1032     38.1599
            2       y2       3SLS          50      38.1599     74.6253
            3       y1       GMM           50      27.4205     46.4028
            4       y2       GMM           50      46.4028     99.4656
```

Figure 20.58. SDATA= Data Set

VDATA= Input data set

The VDATA= option allows a variance matrix for GMM estimation to be input from a data set. When the VDATA= option is used on the PROC MODEL or FIT statement, the matrix that is input is used to define the objective function and is used as the initial V for the methods that iterate the V matrix.

Normally the VDATA= matrix is created from the OUTV= option on a previous FIT statement. Alternately an input VDATA= data set can be created using the DATA step. Each row and column of the V matrix is associated with an equation and an instrument. The position of each element in the V matrix can then be indicated by an equation name and an instrument name for the row of the element and an equation name and an instrument name for the column. Each observation in the VDATA= data set is an element in the V matrix. The row and column of the element are indicated by four variables EQ_ROW, INST_ROW, EQ_COL, and INST_COL which contain the equation name or instrument name. The variable name for an element is VALUE. Missing values are set to 0. Because the variance matrix is symmetric, only a triangular part of the matrix needs to be input.

The following SAS statements are used to generate a **V** matrix estimation from GMM and to store that estimate in the data set GMMV:

```
proc model data=gmm2 ;
   exogenous x1 x2;
   parms a1 a2 b2 b1 2.5 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / gmm outv=gmmv;
run;
```

The data set GMM2 was generated by the example in the preceding ESTDATA= section. The **V** matrix stored in GMMV is selected for use in an additional GMM estimation by the following FIT statement:

```
fit y1 y2 / gmm vdata=gmmv;
run;

proc print data=gmmv(obs=15);
run;
```

A partial listing of the GMMV data set is shown in Figure 20.59. There are a total of 78 observations in this data set. The **V** matrix is 12 by 12 for this example.

```
Obs    _TYPE_    EQ_ROW    EQ_COL    INST_ROW        INST_COL              VALUE

  1     GMM        Y1        Y1      1               1                   1509.59
  2     GMM        Y1        Y1      X1              1                   8257.41
  3     GMM        Y1        Y1      X1              X1                 47956.08
  4     GMM        Y1        Y1      X2              1                   7136.27
  5     GMM        Y1        Y1      X2              X1                 44494.70
  6     GMM        Y1        Y1      X2              X2                153135.59
  7     GMM        Y1        Y1      @PRED.Y1/@B1    1                  47957.10
  8     GMM        Y1        Y1      @PRED.Y1/@B1    X1                289178.68
  9     GMM        Y1        Y1      @PRED.Y1/@B1    X2                275074.36
 10     GMM        Y1        Y1      @PRED.Y1/@B1    @PRED.Y1/@B1     1789176.56
 11     GMM        Y1        Y1      @PRED.Y2/@B2    1                 152885.91
 12     GMM        Y1        Y1      @PRED.Y2/@B2    X1                816886.49
 13     GMM        Y1        Y1      @PRED.Y2/@B2    X2               1121114.96
 14     GMM        Y1        Y1      @PRED.Y2/@B2    @PRED.Y1/@B1     4576643.57
 15     GMM        Y1        Y1      @PRED.Y2/@B2    @PRED.Y2/@B2    28818318.24
```

Figure 20.59. The First 15 Observations in the VDATA= Data Set

Output Data Sets

OUT= Data Set

For normalized form equations, the OUT= data set specified on the FIT statement contains residuals, actuals, and predicted values of the dependent variables computed from the parameter estimates. For general form equations, actual values of the endogenous variables are copied for the residual and predicted values.

The variables in the data set are as follows:

- BY variables
- RANGE variable
- ID variables
- _ESTYPE_, a character variable of length 8 identifying the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- _TYPE_, a character variable of length 8 identifying the type of observation: RESIDUAL, PREDICT, or ACTUAL
- _WEIGHT_, the weight of the observation in the estimation. The _WEIGHT_ value is 0 if the observation was not used. It is equal to the product of the _WEIGHT_ model program variable and the variable named in the WEIGHT statement, if any, or 1 if weights were not used.
- the WEIGHT statement variable if used
- the model variables. The dependent variables for the normalized-form equations in the estimation contain residuals, actuals, or predicted values, depending on the _TYPE_ variable, whereas the model variables that are not associated with estimated equations always contain actual values from the input data set.
- any other variables named in the OUTVARS statement. These can be program variables computed by the model program, CONTROL variables, parameters, or special variables in the model program.

The following SAS statements are used to generate and print an OUT= data set:

```
proc model data=gmm2;
   exogenous x1 x2;
   parms a1 a2 b2 b1 2.5 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm out=resid outall ;
run;

proc print data=resid(obs=20);
run;
```

The data set GMM2 was generated by the example in the preceding ESTDATA= section above. A partial listing of the RESID data set is shown in Figure 20.60.

```
Obs    _ESTYPE_     _TYPE_     _WEIGHT_       x1           x2       y1           y2

  1      3SLS      ACTUAL         1       1.00000     -1.7339   -3.05812    -23.071
  2      3SLS      PREDICT        1       1.00000     -1.7339   -0.36806    -19.351
  3      3SLS      RESIDUAL       1       1.00000     -1.7339   -2.69006     -3.720
  4      3SLS      ACTUAL         1       1.41421     -5.3046    0.59405     43.866
  5      3SLS      PREDICT        1       1.41421     -5.3046   -0.49148     45.588
  6      3SLS      RESIDUAL       1       1.41421     -5.3046    1.08553     -1.722
  7      3SLS      ACTUAL         1       1.73205     -5.2826    3.17651     51.563
  8      3SLS      PREDICT        1       1.73205     -5.2826   -0.48281     41.857
  9      3SLS      RESIDUAL       1       1.73205     -5.2826    3.65933      9.707
 10      3SLS      ACTUAL         1       2.00000     -0.6878    3.66208    -70.011
 11      3SLS      PREDICT        1       2.00000     -0.6878   -0.18592    -76.502
 12      3SLS      RESIDUAL       1       2.00000     -0.6878    3.84800      6.491
 13      3SLS      ACTUAL         1       2.23607     -7.0797    0.29210     99.177
 14      3SLS      PREDICT        1       2.23607     -7.0797   -0.53732     92.201
 15      3SLS      RESIDUAL       1       2.23607     -7.0797    0.82942      6.976
 16      3SLS      ACTUAL         1       2.44949     14.5284    1.86898    423.634
 17      3SLS      PREDICT        1       2.44949     14.5284   -1.23490    421.969
 18      3SLS      RESIDUAL       1       2.44949     14.5284    3.10388      1.665
 19      3SLS      ACTUAL         1       2.64575     -0.6968   -1.03003    -72.214
 20      3SLS      PREDICT        1       2.64575     -0.6968   -0.10353    -69.680
```

Figure 20.60. The OUT= Data Set

OUTEST= Data Set

The OUTEST= data set contains parameter estimates and, if requested, estimates of the covariance of the parameter estimates.

The variables in the data set are as follows:

- BY variables
- _NAME_, a character variable of length 32, blank for observations containing parameter estimates or a parameter name for observations containing covariances
- _TYPE_, a character variable of length 8 identifying the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- the parameters estimated.

If the COVOUT option is specified, an additional observation is written for each row of the estimate of the covariance matrix of parameter estimates, with the _NAME_ values containing the parameter names for the rows. Parameter names longer than 32 characters are truncated.

OUTPARMS= Data Set

The option OUTPARMS= writes all the parameter estimates to an output data set. This output data set contains one observation and is similar to the OUTEST= data set, but it contains all the parameters, is not associated with any FIT task, and contains no covariances. The OUTPARMS= option is used on the PROC MODEL statement, and the data set is written at the end, after any FIT or SOLVE steps have been performed.

OUTS= Data Set

The OUTS= SAS data set contains the estimate of the covariance matrix of the residuals across equations. This matrix is formed from the residuals that are computed using the parameter estimates.

The variables in the OUTS= data set are as follows:

- BY variables
- _NAME_, a character variable containing the name of the equation
- _TYPE_, a character variable of length 8 identifying the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- variables with the names of the equations in the estimation.

Each observation contains a row of the covariance matrix. The data set is suitable for use with the SDATA= option on a subsequent FIT or SOLVE statement. (See "Tests on Parameters" in this chapter for an example of the SDATA= option.)

OUTSUSED= Data Set

The OUTSUSED= SAS data set contains the covariance matrix of the residuals across equations that is used to define the objective function. The form of the OUTSUSED= data set is the same as that for the OUTS= data set.

Note that OUTSUSED= is the same as OUTS= for the estimation methods that iterate the **S** matrix (ITOLS, IT2SLS, ITSUR, and IT3SLS). If the SDATA= option is specified in the FIT statement, OUTSUSED= is the same as the SDATA= matrix read in for the methods that do not iterate the **S** matrix (OLS, SUR, N2SLS, and N3SLS).

OUTV= Data Set

The OUTV= data set contains the estimate of the variance matrix, V. This matrix is formed from the instruments and the residuals that are computed using the parameter estimates obtained from the initial 2SLS estimation when GMM estimation is selected. If an estimation method other than GMM or ITGMM is requested and OUTV= is specified, a V matrix is created using computed estimates. In the case that a VDATA= data set is used, this becomes the OUTV= data set. For ITGMM, the OUTV= data set is the matrix formed from the instruments and the residuals computed using the final parameter estimates.

ODS Table Names

PROC MODEL assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, “Using the Output Delivery System.”

Table 20.3. ODS Tables Produced in PROC MODEL

ODS Table Name	Description	Option
ODS Tables Created by the FIT Statement		
AugGMMCovariance	Cross products matrix	GMM
ChowTest	Structural change test	CHOW=
CollinDiagnostics	Collinearity Diagnostics	
ConfInterval	Profile likelihood Confidence Intervals	PRL=
ConvCrit	Convergence criteria for estimation	default
ConvergenceStatus	Convergence status	default
CorrB	Correlations of parameters	COVB/CORRB
CorrResiduals	Correlations of residuals	CORRS/COVS
CovB	Covariance of parameters	COVB/CORRB
CovResiduals	Covariance of residuals	CORRS/COVS
Crossproducts	Cross products matrix	ITALL/ITPRINT
DatasetOptions	Data sets used	default
DetResidCov	Determinant of the Residuals	DETAILS
DWTest	Durbin Watson Test	DW=
Equations	Listing of equations to estimate	default
EstSummaryMiss	Model Summary Statistics for PAIRWISE	MISSING=
EstSummaryStats	Objective, Objective * N	default
GMMCovariance	Cross products matrix	GMM
Godfrey	Godfrey’s Serial Correlation Test	GF=
HausmanTest	Hausman’s test table	HAUSMAN
HeteroTest	Heteroscedasticity test tables	BREUSCH/PAGEN
InvXPXMat	X’X inverse for System	I
IterInfo	Iteration printing	ITALL/ITPRINT
LagLength	Model lag length	default
MinSummary	Number of parameters, estimation kind	default
MissingValues	Missing values generated by the program	default
ModSummary	Listing of all categorized variables	default
ModVars	Listing of Model variables and parameters	default
NormalityTest	Normality test table	NORMAL
ObsSummary	Identifies observations with errors	default
ObsUsed	Observations read, used, and missing.	default
ParameterEstimates	Parameter Estimates	default
ParmChange	Parameter Change Vector	
ResidSummary	Summary of the SSE, MSE for the equations	default
SizeInfo	Storage Requirement for estimation	DETAILS
TermEstimates	Nonlinear OLS and ITOLS Estimates	OLS/ITOLS
TestResults	Test statement table	
WgtVar	The name of the weight variable	
XPXMat	X’X for System	XPX

Table 20.3. (continued)

ODS Table Name	Description	Option
ODS Tables Created by the SOLVE Statement		
DatasetOptions	Data sets used	default
DescriptiveStatistics	Descriptive Statistics	STATS
FitStatistics	Fit statistics for simulation	STATS
LagLength	Model lag length	default
ModSummary	Listing of all categorized variables	default
ObsSummary	Simulation trace output	SOLVEPRINT
ObsUsed	Observations read, used, and missing.	default
SimulationSummary	Number of variables solved for	default
SolutionVarList	Solution Variable Lists	default
TheilRelStats	Theil Relative Change Error Statistics	THEIL
TheilStats	Theil Forecast Error Statistics	THEIL
ODS Tables Created by the FIT and SOLVE Statements		
AdjacencyMatrix	Adjacency Graph	GRAPH
BlockAnalysis	Block analysis	BLOCK
BlockStructure	Block structure	BLOCK
CodeDependency	Variable cross reference	LISTDEP
CodeList	Listing of programs statements	LISTCODE
CrossReference	Cross Reference Listing For Program	
DepStructure	Dependency Structure of the System	BLOCK
DerList	Derivative variables	LISTDER
FirstDerivatives	First derivative table	LISTDER
InterIntg	Integration Iteration Output	INTGPRINT
MemUsage	Memory usage statistics	MEMORYUSE
ParmReadIn	Parameter estimates read in	ESTDATA=
ProgList	Listing of Compiled Program Code	
RangeInfo	RANGE statement specification	
SortAdjacencyMatrix	Sorted adjacency Graph	GRAPH
TransitiveClosure	Transitive closure Graph	GRAPH

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the MODEL procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

ODS Graph Names

PROC MODEL assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 20.4.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 20.4. ODS Graphics Produced by PROC MODEL

ODS Graph Name	Plot Description
ACFPlot	Autocorrelation of residuals
ActualByPredicted	Predicted vs actual plot
CooksD	Cook's D plot
IACFPlot	Inverse autocorrelation of residuals
QQPlot	QQ plot of residuals
PACFPlot	Partial autocorrelation of residuals
ResidualHistogram	Histogram of the residuals
StudentResidualPlot	Studentized residual plot

Details: Simulation

The *solution* given the vector **k**, of the following nonlinear system of equations is the vector **u** which satisfies this equation:

$$\mathbf{q}(\mathbf{u}, \mathbf{k}, \theta) = 0$$

A *simulation* is a set of solutions $\mathbf{u}_t$ for a specific sequence of vectors $\mathbf{k}_t$.

Model simulation can be performed to

- check how well the model predicts the actual values over the historical period
- investigate the sensitivity of the solution to changes in the input values or parameters
- examine the dynamic characteristics of the model
- check the stability of the simultaneous solution
- estimate the statistical distribution of the predicted values of the nonlinear model using Monte Carlo methods

By combining the various solution modes with different input data sets, model simulation can answer many different questions about the model. This section presents details of model simulation and solution.

Solution Modes

The following solution modes are commonly used:

- *Dynamic simultaneous forecast* mode is used for forecasting with the model. Collect the historical data on the model variables, the future assumptions of the exogenous variables, and any prior information on the future endogenous values, and combine them in a SAS data set. Use the FORECAST option on the SOLVE statement.
- *Dynamic simultaneous simulation* mode is often called *ex-post simulation*, *historical simulation*, or *ex-post forecasting*. Use the DYNAMIC option. This mode is the default.
- *Static simultaneous simulation* mode can be used to examine the within-period performance of the model without the complications of previous period errors. Use the STATIC option.
- *NAHEAD=n dynamic simultaneous simulation* mode can be used to see how well *n*-period-ahead forecasting would have performed over the historical period. Use the NAHEAD=*n* option.

The different solution modes are explained in detail in the following sections.

Dynamic and Static Simulations

In model simulation, either solved values or actual values from the data set can be used to supply lagged values of an endogenous variable. A *dynamic* solution refers to a solution obtained by using only solved values for the lagged values. Dynamic mode is used both for forecasting and for simulating the dynamic properties of the model.

A *static* solution refers to a solution obtained by using the actual values when available for the lagged endogenous values. Static mode is used to simulate the behavior of the model without the complication of previous period errors. Dynamic simulation is the default.

If you wish to use static values for lags only for the first *n* observations, and dynamic values thereafter, specify the START=*n* option. For example, if you want a dynamic simulation to start after observation twenty-four, specify START=24 on the SOLVE statement. If the model being simulated had a value lagged for four time periods, then this value would start using dynamic values when the simulation reached observation number 28.

n-Period-Ahead Forecasting

Suppose you want to regularly forecast 12 months ahead and produce a new forecast each month as more data becomes available. *n*-period-ahead forecasting allows you to test how well you would have done over time had you been using your model to forecast 1 year ahead.

To see how well a model predicts *n* time periods in the future, perform an *n*-period-ahead forecast on real data and compare the forecast values with the actual values.

n-period-ahead forecasting refers to using dynamic values for the lagged endogenous variables only for lags *1* through *n-1*. For example, 1-period-ahead forecasting, specified by the NAHEAD=1 option on the SOLVE statement, is the same as if a static solution had been requested. Specifying NAHEAD=2 produces a solution that uses dynamic values for lag one and static, actual, values for longer lags.

The following example is a 2-year-ahead dynamic simulation. The output is shown in Figure 20.61.

```
data yearly;
   input year x1 x2 x3 y1 y2 y3;
   datalines;
84 4 9  0  7  4  5
85 5 6  1  1  27  4
86 3 8  2  5  8  2
87 2 10 3  0  10 10
88 4 7  6  20 60 40
89 5 4  8  40 40 40
90 3 2  10 50 60 60
91 2 5  11 40 50 60
;
run;

proc model data=yearly outmodel=foo;
   endogenous y1 y2 y3;
   exogenous  x1 x2 x3;

   y1 = 2 + 3*x1 - 2*x2 + 4*x3;
   y2 = 4 + lag2( y3 ) + 2*y1 + x1;
   y3 = lag3( y1 ) + y2 - x2;

   solve y1 y2 y3 / nahead=2 out=c;
run;

proc print data=c;run;
```

```
                         The MODEL Procedure
         Dynamic Simultaneous 2-Periods-Ahead Forecasting Simulation

                           Data Set Options

                           DATA=    YEARLY
                           OUT=     C


                          Solution Summary

                  Variables Solved               3
                  Simulation Lag Length          3
                  Solution Method           NEWTON
                  CONVERGE=                   1E-8
                  Maximum CC                     0
                  Maximum Iterations             1
                  Total Iterations               8
                  Average Iterations             1


                        Observations Processed

                            Read      20
                            Lagged    12
                            Solved     8
                            First      5
                            Last       8


                  Variables Solved For    y1 y2 y3
```

Figure 20.61. NAHEAD Summary Report

```
Obs    _TYPE_      _MODE_     _LAG_    _ERRORS_    y1     y2     y3    x1    x2    x3

  1    PREDICT   SIMULATE     0          0         0     10      7     2    10     3
  2    PREDICT   SIMULATE     1          0        24     58     52     4     7     6
  3    PREDICT   SIMULATE     1          0        41    101    102     5     4     8
  4    PREDICT   SIMULATE     1          0        47    141    139     3     2    10
  5    PREDICT   SIMULATE     1          0        42    130    145     2     5    11
```

Figure 20.62. C Data Set

The preceding 2-year-ahead simulation can be emulated without using the NAHEAD= option by the following PROC MODEL statements:

```
proc model data=test model=foo;
  range year = 87 to 88;
  solve y1 y2 y3 / dynamic solveprint;
run;

  range year = 88 to 89;
  solve y1 y2 y3 / dynamic solveprint;
run;

  range year = 89 to 90;
  solve y1 y2 y3 / dynamic solveprint;
run;
```

```
range year = 90 to 91;
solve y1 y2 y3 / dynamic solveprint;
```

The totals shown under "Observations Processed" in Figure 20.61 are equal to the sum of the four individual runs.

Simulation and Forecasting

You can perform a simulation of your model or use the model to produce forecasts. *Simulation* refers to the determination of the endogenous or dependent variables as a function of the input values of the other variables, even when actual data for some of the solution variables are available in the input data set. The simulation mode is useful for verifying the fit of the model parameters. Simulation is selected by the SIMULATE option on the SOLVE statement. Simulation mode is the default.

In forecast mode, PROC MODEL solves only for those endogenous variables that are missing in the data set. The actual value of an endogenous variable is used as the solution value whenever nonmissing data for it are available in the input data set. Forecasting is selected by the FORECAST option on the SOLVE statement.

For example, an econometric forecasting model can contain an equation to predict future tax rates, but tax rates are usually set in advance by law. Thus, for the first year or so of the forecast, the predicted tax rate should really be exogenous. Or, you may want to use a prior forecast of a certain variable from a short-run forecasting model to provide the predicted values for the earlier periods of a longer-range forecast of a long-run model. A common situation in forecasting is when historical data needed to fill the initial lags of a dynamic model are available for some of the variables but have not yet been obtained for others. In this case, the forecast must start in the past to supply the missing initial lags. Clearly, you should use the actual data that are available for the lags. In all the preceding cases, the forecast should be produced by running the model in the FORECAST mode; simulating the model over the future periods would not be appropriate.

Monte Carlo Simulation

The accuracy of the forecasts produced by PROC MODEL depends on four sources of error (Pindyck 1981, 405-406):

- The system of equations contains an implicit random error term ϵ

  $$\mathbf{g}(\mathbf{y}, \mathbf{x}, \hat{\theta}) = \epsilon$$

  where **y**, **x**, **g**, $\hat{\theta}$, and ϵ are vector valued.
- The estimated values of the parameters, $\hat{\theta}$, are themselves random variables.
- The exogenous variables may have been forecast themselves and therefore may contain errors.
- The system of equations may be incorrectly specified; the model only approximates the process modeled.

The RANDOM= option is used to request Monte Carlo (or stochastic) simulations to generate confidence intervals for errors arising from the first two sources. The Monte Carlo simulations can be performed with ϵ, θ, or both vectors represented as random variables. The SEED= option is used to control the random number generator for the simulations. SEED=0 forces the random number generator to use the system clock as its seed value.

In Monte Carlo simulations, repeated simulations are performed on the model for random perturbations of the parameters and the additive error term. The random perturbations follow a multivariate normal distribution with expected value of 0 and covariance described by a covariance matrix of the parameter estimates in the case of θ, or a covariance matrix of the equation residuals for the case of ϵ. PROC MODEL can generate both covariance matrices or you can provide them.

The ESTDATA= option specifies a data set containing an estimate of the covariance matrix of the parameter estimates to use for computing perturbations of the parameters. The ESTDATA= data set is usually created by the FIT statement with the OUTEST= and OUTCOV options. When the ESTDATA= option is specified, the matrix read from the ESTDATA= data set is used to compute vectors of random shocks or perturbations for the parameters. These random perturbations are computed at the start of each repetition of the solution and added to the parameter values. The perturbed parameters are fixed throughout the solution range. If the covariance matrix of the parameter estimates is not provided, the parameters are not perturbed.

The SDATA= option specifies a data set containing the covariance matrix of the residuals to use for computing perturbations of the equations. The SDATA= data set is usually created by the FIT statement with the OUTS= option. When SDATA= is specified, the matrix read from the SDATA= data set is used to compute vectors of random shocks or perturbations for the equations. These random perturbations are computed at each observation. The simultaneous solution satisfies the model equations plus the random shocks. That is, the solution is not a perturbation of a simultaneous solution of the structural equations; rather, it is a simultaneous solution of the stochastic equations using the simulated errors. If the SDATA= option is not specified, the random shocks are not used.

The different random solutions are identified by the _REP_ variable in the OUT= data set. An unperturbed solution with _REP_=0 is also computed when the RANDOM= option is used. RANDOM=*n* produces *n*+1 solution observations for each input observation in the solution range. If the RANDOM= option is not specified, the SDATA= and ESTDATA= options are ignored, and no Monte Carlo simulation is performed.

PROC MODEL does not have an automatic way of modeling the exogenous variables as random variables for Monte Carlo simulation. If the exogenous variables have been forecast, the error bounds for these variables should be included in the error bounds generated for the endogenous variables. If the models for the exogenous variables are included in PROC MODEL, then the error bounds created from a Monte Carlo simulation will contain the uncertainty due to the exogenous variables.

Alternatively, if the distribution of the exogenous variables is known, the built-in random number generator functions can be used to perturb these variables appropriately for the Monte Carlo simulation. For example, if you knew the forecast of an exogenous variable, X, had a standard error of 5.2 and the error was normally distributed, then the following statements could be used to generate random values for X:

```
   x_new = x + 5.2 * rannor(456);
```

During a Monte Carlo simulation the random number generator functions produce one value at each observation. It is important to use a different seed value for all the random number generator functions in the model program; otherwise, the perturbations will be correlated. For the unperturbed solution, _REP_=0, the random number generator functions return 0.

PROC UNIVARIATE can be used to create confidence intervals for the simulation (see the Monte Carlo simulation example in the "Getting Started" section).

Multivariate t-Distribution Simulation

To perform a Monte Carlo analysis of models that have residuals distributed as a multivariate *t*, use the ERRORMODEL statement with either the $\sim$ t(*variance*, *df*) option or with the CDF=t(*variance*, *df*) option. The CDF= option specifies the distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another.

The following is an example of estimating and simulating a system of equations with *t*-distributed errors using the ERRORMODEL statement:

```
       /* generate simulation data set */
data five;
set xfrate end=last;
if last then do;
```

```
    todate = date +5;
    do date = date to todate;
       output;
    end;
 end;
```

The preceding DATA step generates the data set to request a five-days-ahead forecast. The following statements estimate and forecast the three forward-rate models of the following form.

$$\begin{aligned} rate_t &= rate_{t-1} + \mu * rate_{t-1} + \nu \\ \nu &= \sigma * rate_{t-1} * \epsilon \\ \epsilon &\sim \mathrm{N}(0,1) \end{aligned}$$

```
Title "Daily Multivariate Geometric Brownian Motion Model "
         "of D-Mark/USDollar Forward Rates";

proc model data=xfrate;

   parms df 15;       /* Give initial value to df */

   demusd1m = lag(demusd1m) + mu1m * lag(demusd1m);
   var_demusd1m = sigma1m ** 2 * lag(demusd1m **2);
   demusd3m = lag(demusd3m) + mu3m * lag(demusd3m);
   var_demusd3m = sigma3m ** 2 * lag(demusd3m ** 2);
   demusd6m = lag(demusd6m) + mu6m * lag(demusd6m);
   var_demusd6m = sigma6m ** 2 * lag(demusd6m ** 2);

      /* Specify the error distribution */
   errormodel demusd1m demusd3m demusd6m
        ~ t( var_demusd1m var_demusd3m var_demusd6m, df );

      /* output normalized S matrix */
   fit demusd1m demusd3m demusd6m / outsn=s;
run;
      /* forecast five days in advance */
   solve demusd1m demusd3m demusd6m /
         data=five sdata=s random=1500 out=monte;
   id date;
run;

   /* select out the last date ---*/
data monte; set monte;
   if date = ’10dec95’d then output;
run;

title "Distribution of demusd1m Five Days Ahead";
proc univariate data=monte noprint;
     var demusd1m;
     histogram demusd1m / normal(noprint color=red)
```

```
          kernel(noprint color=blue) cfill=ligr;
run;
```

The Monte Carlo simulation specified in the preceding example draws from a multivariate *t*-distribution with constant degrees of freedom and forecasted variance and computes future states of DEMUSD1M, DEMUSD3M, and DEMUSD6M. The OUTSN= option on the FIT statement is used to specify the data set for the normalized Σ matrix. That is the Σ matrix is created by crossing the normally distributed residuals. The normally distributed residuals are created from the *t*-distributed residuals using the normal inverse CDF and the *t* CDF. This matrix is a correlation matrix.

The distribution of DEMUSD1M on the fifth day is shown in the following output. The two curves overlayed on the graph are a kernel density estimation and a normal distribution fit to the results.

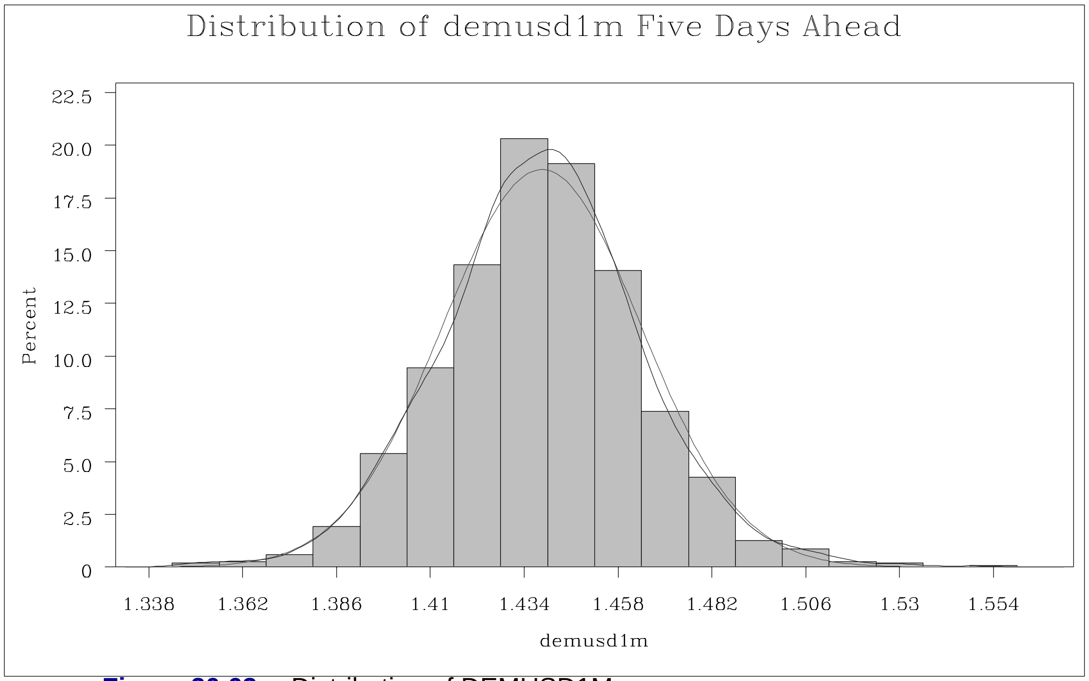


Figure 20.63. Distribution of DEMUSD1M

Alternate Distribution Simulation

As an alternate to the normal distribution, the ERRORMODEL statement can be used in a simulation to specify other distributions. The distributions available for simulation are Cauchy, Chi-squared, *F*, Poisson, *t*, and Uniform. An empirical distribution can also be used if the residuals are specified using the RESIDDATA= option on the SOLVE statement.

Except for the *t*, all of these alternate distributions are univariate but can be used together in a multivariate simulation. The ERRORMODEL statement applies to solved for equations only. That is, the normal form or general form equation referred to by the ERRORMODEL statement must be one of the equations you have selected in the SOLVE statement.

In the following example, two Poisson distributed variables are used to simulate the calls arriving and leaving a call center.

```
data s;    /* Covariance between arriving and leaving */
   arriving = 1; leaving = 0.7; _name_= "arriving";
   output;
   arriving = 0.7; leaving = 1.0; _name_= "leaving";
   output;
run;

data calls;
   date = '20mar2001'd;
   output;
run;
```

The first DATA step generates a data set containing a covariance matrix for the ARRIVING and LEAVING variables. The covariance is

$$\begin{vmatrix} 1 & .7 \\ .7 & 1 \end{vmatrix}$$

```
proc model data=calls;
   arriving = 10;
   errormodel arriving ~ poisson( 10 );

      /* Have four people answering the phone */
   leaving  = 4;
   errormodel leaving ~ poisson( 11 );

   waiting = arriving - leaving;

   solve arriving leaving / random=500 sdata=s out=sim;
run;

title "Distribution of Clients Waiting";
proc univariate data=sim noprint;
     var waiting ;
     histogram waiting / cfill=ligr;
run;
```

The distribution of number of waiting clients is shown in the following output.

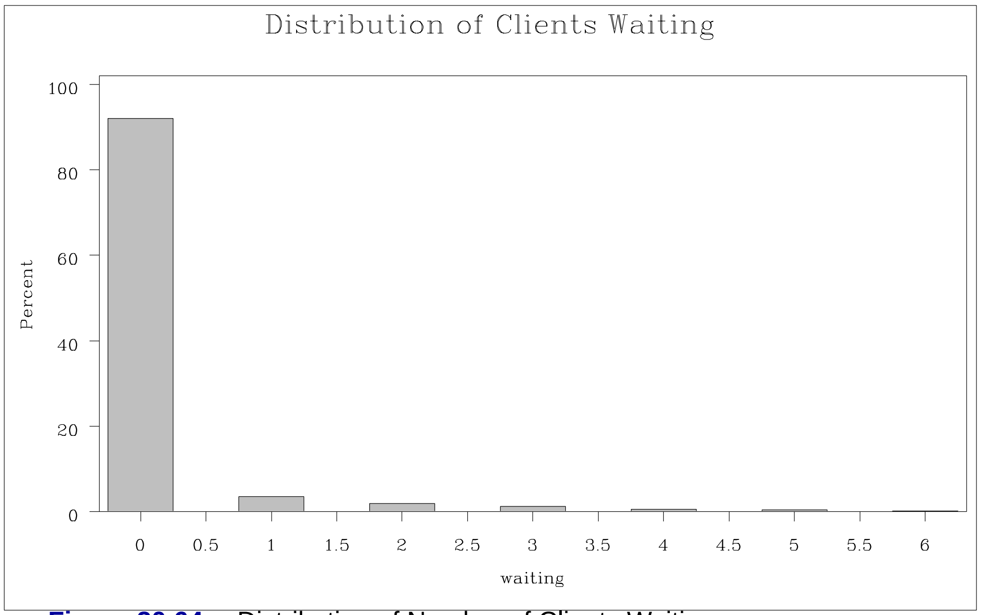


Figure 20.64. Distribution of Number of Clients Waiting

Mixtures of Distributions - Copulas

The theory of copulas is what enables the MODEL procedure to combine and simulate multivariate distributions with different marginals. This section provides a brief overview of copulas.

Modeling a system of variables accurately is a difficult task. The underlying, ideal, distributional assumptions for each variable are usually different from each other. An individual variable may be best modeled as a *t*-distribution or as a Poisson process. The correlation of the various variables are very important to estimate as well. A joint estimation of a set of variables would make it possible to estimate a correlation structure but would restrict the modeling to single, simple multivariate distribution (for example, the norma l). Even with a simple multivariate distribution, the joint estimation would be computationally difficult and would have to deal with issues of missing data.

Using the MODEL procedure ERRORMODEL statement you can combine and simulate from models of different distributions. The covariance matrix for the combined model is constructed using the copula induced by the multivariate normal distribution. A copula is a function that couples joint distributions to their marginal distributions.

The copula used by the model procedure is based on the multivariate normal. This particular multivariate normal has zero mean and covariance matrix R. The user provides R, which can be created using the following steps

1. Each model is estimated separately and their residuals saved.
2. The residuals for each model are converted to a normal distribution using their CDFs, $F_i(.)$, using the relationship $\Phi^{-1}(F(\epsilon_{it}))$.
3. Cross these normal residuals, to create a covariance matrix R.

If the model of interest can be estimated jointly, such as multivariate T, then the OUTSN= option can be used to generate the correct covariance matrix.

A draw from this mixture of distributions is created using the following steps that are performed automatically by the MODEL procedure.

1. Independent $N(0, 1)$ variables are generated.
2. These variables are transformed to a correlated set using the covariance matrix R.
3. These correlated normals are transformed to a uniform using $\Phi()$.
4. $F^{-1}()$ is used to compute the final sample value.

Quasi-Random Number Generators

Traditionally high discrepancy pseudo-random number generators are used to generate innovations in Monte Carlo simulations. Loosely translated, a high discrepancy pseudo-random number generator is one in which there is very little correlation between the current number generated and the past numbers generated. This property is ideal if indeed independence of the innovations is required. If, on the other hand, the efficient spanning of a multi-dimensional space is desired, a low discrepancy, quasi-random number generator can be used. A quasi-random number generator produces numbers which have no random component.

A simple one-dimensional quasi-random sequence is the van der Corput sequence. Given a prime number r ($r \geq 2$) any integer has a unique representation in terms of base r. A number in the interval [0,1) can be created by inverting the representation base power by base power. For example, consider r=3 and n=1. 1 in base 3 is

$$1_{10} = 1 \cdot 3^0 = 1_3$$

When the powers of 3 are inverted,

$$\phi(1) = \frac{1}{3}$$

Also 11 in base 3 is

$$11_{10} = 1 \cdot 3^2 + 2 \cdot 3^0 = 102_3$$

When the powers of 3 are inverted,

$$\phi(11) = \frac{1}{9} + 2 \cdot \frac{1}{3} = \frac{7}{9}$$

The first 10 numbers in this sequence $\phi(1) \ldots \phi(10)$ are provided below

$$0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}$$

As the sequence proceeds it fills in the gaps in a uniform fashion.

Several authors have expanded this idea to many dimensions. Two versions supported by the MODEL procedure are the Sobol sequence (QUASI=SOBOL) and the Faure sequence (QUASI=FAURE). The Sobol sequence is based on binary numbers an is generally computationally faster than the Faure sequence. The Faure sequence uses the dimensionality of the problem to determine the number base to use to generate the sequence. The Faure sequence has better distributional properties than the Sobol sequence for dimensions greater than 8.

As an example of the difference between a pseudo random number and a quasi random number consider simulating a bivariate normal with 100 draws.

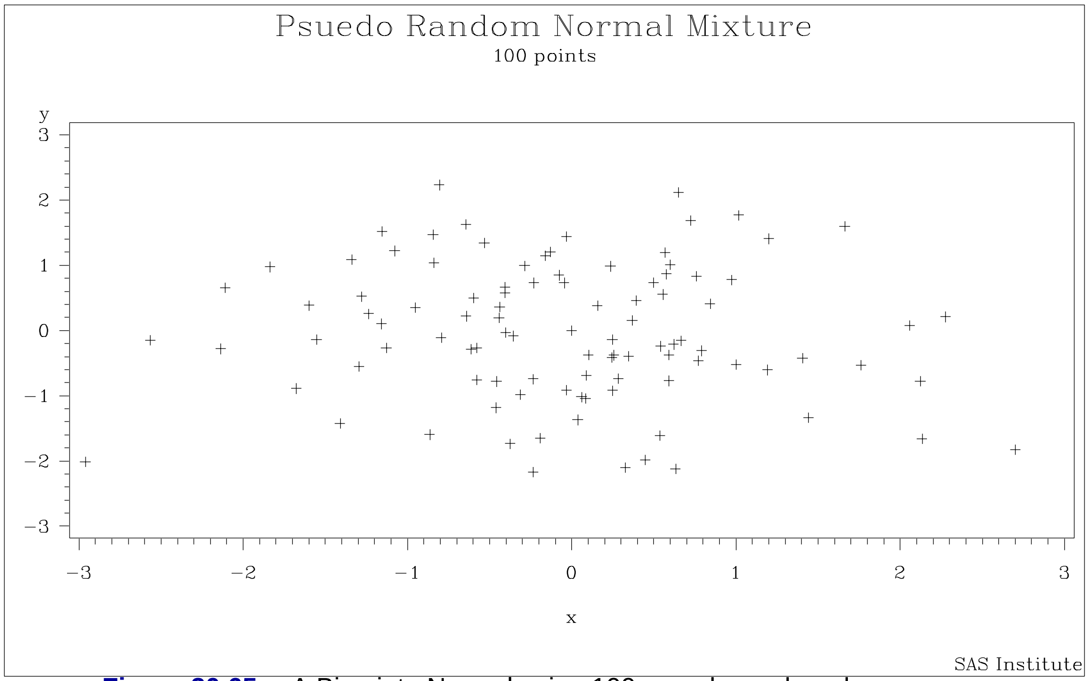


Figure 20.65. A Bivariate Normal using 100 pseudo random draws

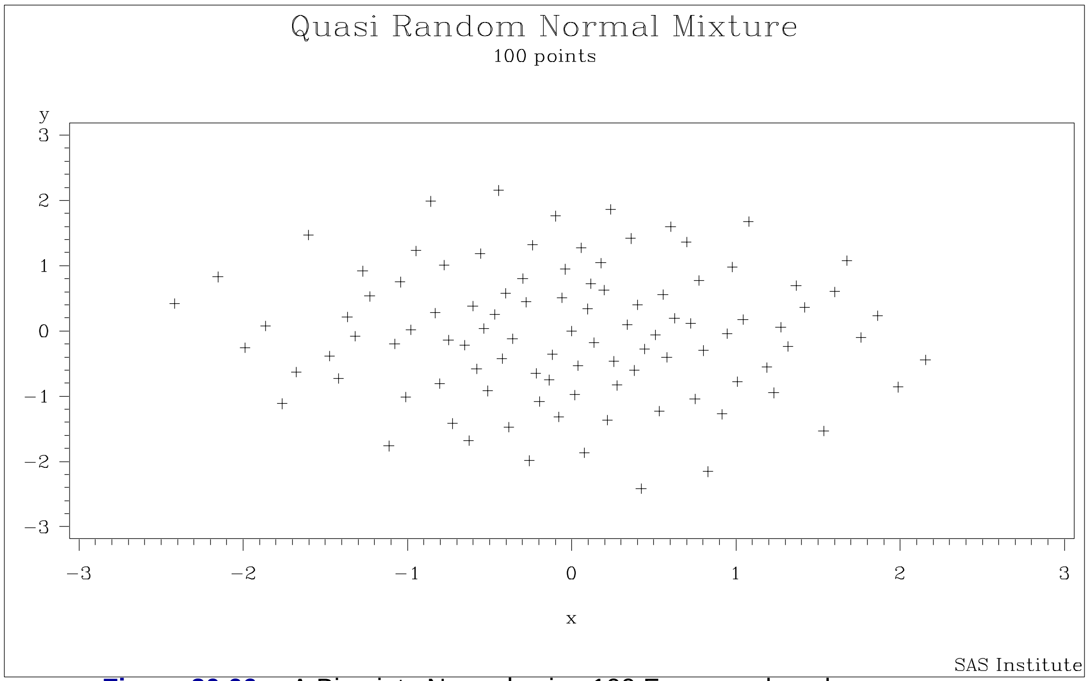


Figure 20.66. A Bivariate Normal using 100 Faure random draws

Solution Mode Output

The following SAS statements dynamically forecast the solution to a nonlinear equation:

```
proc model data=sashelp.citimon;
   parameters a 0.010708  b  -0.478849 c 0.929304;
   lhur = 1/(a * ip) + b + c * lag(lhur);
   solve lhur / out=sim forecast dynamic;
run;
```

The first page of output produced by the SOLVE step is shown in Figure 20.67. This is the summary description of the model. The error message states that the simulation was aborted at observation 144 because of missing input values.

```
                         The MODEL Procedure

                            Model Summary

                      Model Variables         1
                      Parameters              3
                      Equations               1
                      Number of Statements    1
                      Program Lag Length      1


             Model Variables  LHUR
                  Parameters  a(0.010708) b(-0.478849) c(0.929304)
                   Equations  LHUR



                         The MODEL Procedure
                   Dynamic Single-Equation Forecast

ERROR: Solution values are missing because of missing input values for
       observation 144 at NEWTON iteration 0.
NOTE: Additional information on the values of the variables at this
      observation, which may be helpful in determining the cause of the failure
      of the solution process, is printed below.
Iteration Errors - Missing.
NOTE: Simulation aborted.
```

Figure 20.67. Solve Step Summary Output

The second page of output, shown in Figure 20.68, gives more information on the failed observation.

```
                         The MODEL Procedure
                   Dynamic Single-Equation Forecast

ERROR: Solution values are missing because of missing input values for
       observation 144 at NEWTON iteration 0.
NOTE: Additional information on the values of the variables at this
      observation, which may be helpful in determining the cause of the failure
      of the solution process, is printed below.


            Observation    144    Iteration    0    CC    -1.000000
                                  Missing      1
Iteration Errors - Missing.


                     --- Listing of Program Data Vector ---
_N_:                144     ACTUAL.LHUR:          .     ERROR.LHUR:           .
IP:                   .     LHUR:           7.10000     PRED.LHUR:            .
RESID.LHUR:           .     a:              0.01071     b:             -0.47885
c:              0.92930

NOTE: Simulation aborted.
```

Figure 20.68. Solve Step Error Message

From the program data vector you can see the variable IP is missing for observation 144. LHUR could not be computed so the simulation aborted.

The solution summary table is shown in Figure 20.69.

```
                     The MODEL Procedure
               Dynamic Single-Equation Forecast

                       Data Set Options

                   DATA=    SASHELP.CITIMON
                   OUT=     SIM


                      Solution Summary

               Variables Solved              1
               Forecast Lag Length           1
               Solution Method          NEWTON
               CONVERGE=                  1E-8
               Maximum CC                    0
               Maximum Iterations            1
               Total Iterations            143
               Average Iterations            1


                    Observations Processed

                        Read      145
                        Lagged      1
                        Solved    143
                        First       2
                        Last      145
                        Failed      1


                Variables Solved For    LHUR
```

Figure 20.69. Solution Summary Report

This solution summary table includes the names of the input data set and the output data set followed by a description of the model. The table also indicates the solution method defaulted to Newton's method. The remaining output is defined as follows.

Maximum CC	is the maximum convergence value accepted by the Newton procedure. This number is always less than the value for "CONVERGE=."
Maximum Iterations	is the maximum number of Newton iterations performed at each observation and each replication of Monte Carlo simulations.
Total Iterations	is the sum of the number of iterations required for each observation and each Monte Carlo simulation.
Average Iterations	is the average number of Newton iterations required to solve the system at each step.
Solved	is the number of observations used times the number of random replications selected plus one, for Monte Carlo simulations. The one additional simulation is the original unperturbed solution. For simulations not involving Monte Carlo, this number is the number of observations used.

Summary Statistics

The STATS and THEIL options are used to select goodness of fit statistics. Actual values must be provided in the input data set for these statistics to be printed. When the RANDOM= option is specified, the statistics do not include the unperturbed (_REP_=0) solution.

STATS Option Output

If the STATS and THEIL options are added to the model in the previous section

```
proc model data=sashelp.citimon;
   parameters a 0.010708  b  -0.478849 c 0.929304;
   lhur= 1/(a * ip) + b + c * lag(lhur) ;
   solve lhur / out=sim dynamic stats theil;
   range date to ’01nov91’d;
run;
```

the STATS output in Figure 20.70 and the THEIL output in Figure 20.71 are generated.

```
                         The MODEL Procedure
                  Dynamic Single-Equation Simulation

                 Solution Range DATE = FEB1980 To NOV1991

                          Descriptive Statistics

                                          Actual            Predicted
  Variable       N Obs       N       Mean     Std Dev       Mean     Std Dev

  LHUR             142     142     7.0887      1.4509     7.2473      1.1465


                            Statistics of fit

                            Mean     Mean %  Mean Abs  Mean Abs        RMS      RMS %
Variable              N     Error     Error     Error   % Error      Error      Error

LHUR                142    0.1585    3.5289    0.6937   10.0001     0.7854    11.2452

                            Statistics of fit

                 Variable       R-Square    Label

                 LHUR             0.7049    UNEMPLOYMENT RATE:
                                            ALL WORKERS,
                                            16 YEARS
```

Figure 20.70. STATS Output

The number of observations (Nobs), the number of observations with both predicted and actual values nonmissing (N), and the mean and standard deviation of the actual and predicted values of the determined variables are printed first. The next set of columns in the output are defined as follows.

Mean Error	$\frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)$
Mean % Error	$\frac{100}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)/y_j$
Mean Abs Error	$\frac{1}{N}\sum_{j=1}^{N}\lvert\hat{y}_j - y_j\rvert$
Mean Abs % Error	$\frac{100}{N}\sum_{j=1}^{N}\lvert(\hat{y}_j - y_j)/y_j\rvert$
RMS Error	$\sqrt{\frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)^2}$
RMS % Error	$100\sqrt{\frac{1}{N}\sum_{j=1}^{N}((\hat{y}_j - y_j)/y_j)^2}$
R-square	$1 - SSE/CSSA$
SSE	$\sum_{j=1}^{N}(\hat{y}_j - y_j)^2$
SSA	$\sum_{j=1}^{N}(y_j)^2$
CSSA	$SSA - \left(\sum_{j=1}^{N} y_j\right)^2$
$\hat{y}$	predicted value
y	actual value

When the RANDOM= option is specified, the statistics do not include the unperturbed (_REP_=0) solution.

THEIL Option Output

The THEIL option specifies that Theil forecast error statistics be computed for the actual and predicted values and for the relative changes from lagged values. Mathematically, the quantities are

$$\hat{y}c = (\hat{y} - lag(y))/lag(y)$$

$$yc = (y - lag(y))/lag(y)$$

where $\hat{y}c$ is the relative change for the predicted value and *yc* is the relative change for the actual value.

```
                               The MODEL Procedure
                        Dynamic Single-Equation Simulation

                     Solution Range DATE = FEB1980 To NOV1991

                         Theil Forecast Error Statistics

                                             MSE Decomposition Proportions
                                        Corr    Bias     Reg    Dist     Var   Covar
Variable          N       MSE      (R)    (UM)    (UR)    (UD)    (US)    (UC)

LHUR          142.0    0.6168     0.85    0.04    0.01    0.95    0.15    0.81

                         Theil Forecast Error Statistics

                           Inequality Coef
          Variable            U1          U    Label

          LHUR            0.1086      0.0539    UNEMPLOYMENT RATE:
                                                ALL WORKERS,
                                                16 YEARS


                 Theil Relative Change Forecast Error Statistics

                  Relative Change           MSE Decomposition Proportions
                                        Corr    Bias     Reg    Dist     Var   Covar
Variable          N       MSE      (R)    (UM)    (UR)    (UD)    (US)    (UC)

LHUR          142.0    0.0126    -0.08    0.09    0.85    0.06    0.43    0.47

                 Theil Relative Change Forecast Error Statistics

                           Inequality Coef
          Variable            U1          U    Label

          LHUR            4.1226      0.8348    UNEMPLOYMENT RATE:
                                                ALL WORKERS,
                                                16 YEARS
```

Figure 20.71. THEIL Output

The columns have the following meaning:

Corr (R) is the correlation coefficient, ρ, between the actual and predicted values.

$$\rho = \frac{\mathrm{cov}(\mathrm{y}, \hat{\mathrm{y}})}{\sigma_a \sigma_p}$$

where σ_p and σ_a are the standard deviations of the predicted and actual values.

Bias (UM) is an indication of systematic error and measures the extent to which the average values of the actual and predicted deviate from each other.

$$\frac{(\mathrm{E}(\mathrm{y}) - \mathrm{E}(\hat{\mathrm{y}}))^2}{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}$$

Reg (UR) is defined as $(\sigma_p - \rho * \sigma_a)^2/MSE$. Consider the regression

$$y = \alpha + \beta\hat{y}$$

If $\hat{\beta} = 1$, UR will equal zero.

Dist (UD) is defined as $(1 - \rho^2)\sigma_a\sigma_a/MSE$ and represents the variance of the residuals obtained by regressing *yc* on $\hat{y}c$.

Var (US) is the variance proportion. US indicates the ability of the model to replicate the degree of variability in the endogenous variable.

$$US = \frac{(\sigma_p - \sigma_a)^2}{MSE}$$

Covar (UC) represents the remaining error after deviations from average values and average variabilities have been accounted for.

$$UC = \frac{2(1-\rho)\sigma_p\sigma_a}{MSE}$$

U1 is a statistic measuring the accuracy of a forecast.

$$U1 = \frac{\sqrt{MSE}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t)^2}}$$

U is the Theil's inequality coefficient defined as follows:

$$U = \frac{\sqrt{MSE}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t)^2} + \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t)^2}}$$

MSE is the mean square error. In the case of the Relative Change Theil statistics, the MSE is computed as follows:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}c_t - yc_t)^2$$

More information on these statistics can be found in the references Maddala (1977, 344–347) and Pindyck and Rubinfeld (1981, 364–365).

Goal Seeking: Solving for Right-Hand-Side Variables

The process of computing input values needed to produce target results is often called *goal seeking*. To compute a goal-seeking solution, use a SOLVE statement that lists the variables you want to solve for and provide a data set containing values for the remaining variables.

Consider the following demand model for packaged rice

$$quantity\ demanded = \alpha_1 + \alpha_2 price^{2/3} + \alpha_3 income$$

where *price* is the price of the package and *income* is disposable personal income. The only variable the company has control over is the price it charges for rice. This model is estimated using the following simulated data and PROC MODEL statements:

```
data demand;
   do t=1 to 40;
       price = (rannor(10) +5) * 10;
       income = 8000 * t ** (1/8);
       demand = 7200 - 1054 * price ** (2/3) +
                7 * income + 100 * rannor(1);
       output;
   end;
run;

data goal;
    demand = 85000;
    income = 12686;
run;
```

The goal is to find the price the company would have to charge to meet a sales target of 85,000 units. To do this, a data set is created with a DEMAND variable set to 85000 and with an INCOME variable set to 12686, the last income value.

```
proc model data=demand ;
   demand = a1 - a2 * price ** (2/3) + a3 * income;
   fit demand / outest=demest;
run;
```

The desired price is then determined using the following PROC MODEL statement:

```
    solve price / estdata=demest data=goal solveprint;
run;
```

The SOLVEPRINT option prints the solution values, number of iterations, and final residuals at each observation. The SOLVEPRINT output from this solve is shown in Figure 20.72.

```
                        The MODEL Procedure
                     Single-Equation Simulation

Observation   1   Iterations   6   CC    0.000000   ERROR.demand    0.000000


                          Solution Values

                               price

                             33.59016
```

Figure 20.72. Goal Seeking, SOLVEPRINT Output

The output indicates that it took 6 Newton iterations to determine the PRICE of 33.5902, which makes the DEMAND value within 16E-11 of the goal of 85,000 units.

Consider a more ambitious goal of 100,000 units. The output shown in Figure 20.73 indicates that the sales target of 100,000 units is not attainable according to this model.

```
                                  The MODEL Procedure
                               Single-Equation Simulation

NOTE: 3 parameter estimates were read from the ESTDATA=DEMEST data set.



                                  The MODEL Procedure
                               Single-Equation Simulation

ERROR: Could not reduce norm of residuals in 10 subiterations.

ERROR: The solution failed because 1 equations are missing or have extreme
       values for observation 1 at NEWTON iteration 1.
NOTE: Additional information on the values of the variables at this
      observation, which may be helpful in determining the cause of the failure
      of the solution process, is printed below.


             Observation    1    Iteration    1    CC    -1.000000
                                 Missing      1
Iteration Errors - Missing.


             Observation    1    Iteration    1    CC    -1.000000
                                 Missing      1
ERROR: 2 execution errors for this observation
NOTE: Check for missing input data or uninitialized lags.
      (Note that the LAG and DIF functions return missing values for the
initial lag starting observations. This is a change from the 1982 and earlier
versions of SAS/ETS which returned zero for uninitialized lags.)
NOTE: Simulation aborted.
```

Figure 20.73. Goal Seeking, Convergence Failure

The program data vector indicates that even with PRICE nearly 0 (4.462312E-22) the demand is still 4,164 less than the goal. You may need to reformulate your model or collect more data to more accurately reflect the market response.

Numerical Solution Methods

If the SINGLE option is not used, PROC MODEL computes values that simultaneously satisfy the model equations for the variables named in the SOLVE statement. PROC MODEL provides three iterative methods, Newton, Jacobi, and Seidel, for computing a simultaneous solution of the system of nonlinear equations.

Single-Equation Solution

For normalized-form equation systems, the solution can either simultaneously satisfy all the equations or can be computed for each equation separately, using the actual values of the solution variables in the current period to compute each predicted value.

By default, PROC MODEL computes a simultaneous solution. The SINGLE option on the SOLVE statement selects single-equation solutions.

Single-equation simulations are often made to produce residuals (which estimate the random terms of the stochastic equations) rather than the predicted values themselves. If the input data and range are the same as that used for parameter estimation, a static single-equation simulation will reproduce the residuals of the estimation.

Newton's Method

The NEWTON option on the SOLVE statement requests Newton's method to simultaneously solve the equations for each observation. Newton's method is the default solution method. Newton's method is an iterative scheme that uses the derivatives of the equations with respect to the solution variables, *J*, to compute a change vector as

$$\Delta \mathbf{y}^i = J^{-1}\mathbf{q}(\mathbf{y}^i, \mathbf{x}, \theta)$$

PROC MODEL builds and solves *J* using efficient sparse matrix techniques. The solution variables $\mathbf{y}^i$ at the *i*th iteration are then updated as

$$\mathbf{y}^{i+1} = \mathbf{y}^i + d \times \Delta \mathbf{y}^i$$

d is a damping factor between 0 and 1 chosen iteratively so that

$$\|\mathbf{q}(\mathbf{y}^{i+1}, \mathbf{x}, \theta)\| < \|\mathbf{q}(\mathbf{y}^i, \mathbf{x}, \theta)\|$$

The number of subiterations allowed for finding a suitable *d* is controlled by the MAXSUBITER= option. The number of iterations of Newton's method allowed for each observation is controlled by MAXITER= option. Refer to Ortega and Rheinbolt (1970) for more details.

Jacobi Method

The JACOBI option on the SOLVE statement selects a matrix-free alternative to Newton's method. This method is the traditional nonlinear Jacobi method found in the literature. The Jacobi method as implemented in PROC MODEL substitutes predicted values for the endogenous variables and iterates until a fixed point is reached. Then necessary derivatives are computed only for the diagonal elements of the jacobian, **J**.

If the normalized-form equation is

$$\mathbf{y} = \mathbf{f}(\mathbf{y}, \mathbf{x}, \theta)$$

the Jacobi iteration has the form

$$\mathbf{y}^{i+1} = \mathbf{f}(\mathbf{y}^i, \mathbf{x}, \theta)$$

Seidel Method

The Seidel method is an order-dependent alternative to the Jacobi method. The Seidel method is selected by the SEIDEL option on the SOLVE statement. The Seidel method is like the Jacobi method except that in the Seidel method the model is further edited to substitute the predicted values into the solution variables immediately after they are computed. Seidel thus differs from the other methods in that the values of the solution variables are not fixed within an iteration. With the other methods, the order of the equations in the model program makes no difference, but the Seidel method may work much differently when the equations are specified in a different sequence. Note that this fixed point method is the traditional nonlinear Seidel method found in the literature.

The iteration has the form

$$\mathbf{y}_j^{i+1} = \mathbf{f}(\hat{\mathbf{y}}^i, \mathbf{x}, \theta)$$

where $\mathbf{y}_j^{i+1}$ is the *j*th equation variable at the *i*th iteration and

$$\hat{\mathbf{y}}^i = (y_1^{i+1}, y_2^{i+1}, y_3^{i+1}, \ldots, y_{j-1}^{i+1}, y_j^i, y_{j+1}^i, \ldots, y_g^i)'$$

If the model is recursive, and if the equations are in recursive order, the Seidel method will converge at once. If the model is block-recursive, the Seidel method may converge faster if the equations are grouped by block and the blocks are placed in block-recursive order. The BLOCK option can be used to determine the block-recursive form.

Jacobi and Seidel Methods with General Form Equations

Jacobi and Seidel solution methods support general form equations.

There are two cases where derivatives are (automatically) computed. The first case is for equations with the solution variable on the right-hand side and on the left-hand side of the equation

$$y^i = f(\mathbf{x}, y^i)$$

In this case the derivative of ERROR.y with respect to y is computed, and the new y approximation is computed as

$$y^{i+1} = y^i - \frac{f(\mathbf{x}, y^i) - y^i}{\partial(f(\mathbf{x}, y^i) - y^i)/\partial y}$$

The second case is a system of equations containing one or more EQ.var equations. In this case, a heuristic algorithm is used to make the assignment of a unique solution variable to each general form equation. Use the DETAILS option on the SOLVE statement to print a listing of the assigned variables.

Once the assignment is made, the new y approximation is computed as

$$y^{i+1} = y^i - \frac{f(\mathbf{x}, \mathbf{y}^i) - y^i}{\partial(f(\mathbf{x}, \mathbf{y}^i) - y^i)/\partial y}$$

If k is the number of general form equations, then k derivatives are required.

The convergence properties of the Jacobi and Seidel solution methods remain significantly poorer than the default Newton's method.

Comparison of Methods

Newton's method is the default and should work better than the others for most small- to medium-sized models. The Seidel method is always faster than the Jacobi for recursive models with equations in recursive order. For very large models and some highly nonlinear smaller models, the Jacobi or Seidel methods can sometimes be faster. Newton's method uses more memory than the Jacobi or Seidel methods.

Both the Newton's method and the Jacobi method are order-invariant in the sense that the order in which equations are specified in the model program has no effect on the operation of the iterative solution process. In order-invariant methods, the values of the solution variables are fixed for the entire execution of the model program. Assignments to model variables are automatically changed to assignments to corresponding equation variables. Only after the model program has completed execution are the results used to compute the new solution values for the next iteration.

Troubleshooting Problems

In solving a simultaneous nonlinear dynamic model you may encounter some of the following problems.

Missing Values

For SOLVE tasks, there can be no missing parameter values. If there are missing right-hand-side variables, this will result in a missing left-hand-side variable for that observation.

Unstable Solutions

A solution may exist but be unstable. An unstable system can cause the Jacobi and Seidel methods to diverge.

Explosive Dynamic Systems

A model may have well-behaved solutions at each observation but be dynamically unstable. The solution may oscillate wildly or grow rapidly with time.

Propagation of Errors

During the solution process, solution variables can take on values that cause computational errors. For example, a solution variable that appears in a LOG function may be positive at the solution but may be given a negative value during one of the iterations. When computational errors occur, missing values are generated and propagated, and the solution process may collapse.

Convergence Problems

The following items can cause convergence problems:

- illegal function values (that is $\sqrt{-1}$)
- local minima in the model equation
- no solution exists
- multiple solutions exist
- initial values too far from the solution
- the CONVERGE= value too small.

When PROC MODEL fails to find a solution to the system, the current iteration information and the program data vector are printed. The simulation halts if actual values are not available for the simulation to proceed. Consider the following program:

```
data test1;
   do t=1 to 50;
      x1 = sqrt(t) ;
      y = .;
      output;
   end;

proc model data=test1;
   exogenous x1 ;
   control a1 -1 b1 -29 c1 -4 ;
   y = a1 * sqrt(y) + b1 * x1 * x1 + c1 * lag(x1);
   solve y / out=sim forecast dynamic ;
run;
```

which produces the output shown in Figure 20.74.

```
                                  The MODEL Procedure
                            Dynamic Single-Equation Forecast

ERROR: Could not reduce norm of residuals in 10 subiterations.

ERROR: The solution failed because 1 equations are missing or have extreme
       values for observation 1 at NEWTON iteration 1.
NOTE: Additional information on the values of the variables at this
      observation, which may be helpful in determining the cause of the failure
      of the solution process, is printed below.


             Observation    1    Iteration    1    CC    -1.000000
                                 Missing      1
Iteration Errors - Missing.


                     --- Listing of Program Data Vector ---
   _N_:               12     ACTUAL.x1:    1.41421     ACTUAL.y:           .
   ERROR.y:            .     PRED.y:             .     RESID.y:            .
   a1:                -1     b1:               -29     c1:                -4
   x1:           1.41421     y:           -0.00109
   @PRED.y/@y:           .   @ERROR.y/@y:          .



             Observation    1    Iteration    1    CC    -1.000000
                                 Missing      1
ERROR: 1 execution errors for this observation
NOTE: Check for missing input data or uninitialized lags.
      (Note that the LAG and DIF functions return missing values for the
initial lag starting observations. This is a change from the 1982 and earlier
versions of SAS/ETS which returned zero for uninitialized lags.)
NOTE: Simulation aborted.
```

Figure 20.74. SOLVE Convergence Problems

At the first observation the following equation is attempted to be solved:

$$y = -\sqrt{y} - 62$$

There is no solution to this problem. The iterative solution process got as close as it could to making Y negative while still being able to evaluate the model. This problem can be avoided in this case by altering the equation.

In other models, the problem of missing values can be avoided by either altering the data set to provide better starting values for the solution variables or by altering the equations.

You should be aware that, in general, a nonlinear system can have any number of solutions, and the solution found may not be the one that you want. When multiple solutions exist, the solution that is found is usually determined by the starting values for the iterations. If the value from the input data set for a solution variable is missing, the starting value for it is taken from the solution of the last period (if nonmissing) or else the solution estimate is started at 0.

Iteration Output

The iteration output, produced by the ITPRINT option, is useful in determining the cause of a convergence problem. The ITPRINT option forces the printing of the solution approximation and equation errors at each iteration for each observation. A portion of the ITPRINT output from the following statement is shown in Figure 20.75.

```
proc model data=test1;
   exogenous x1 ;
   control a1 -1 b1 -29 c1 -4 ;
   y = a1 * sqrt(abs(y)) + b1 * x1 * x1 + c1 * lag(x1);
   solve y / out=sim forecast dynamic itprint;
run;
```

For each iteration, the equation with the largest error is listed in parentheses after the Newton convergence criteria measure. From this output you can determine which equation or equations in the system are not converging well.

```
                        The MODEL Procedure
                  Dynamic Single-Equation Forecast

Observation    1    Iteration    0    CC    613961.39    ERROR.y    -62.01010


                          Predicted Values

                                  y

                             0.0001000


                          Iteration Errors

                                  y

                             -62.01010

Observation    1    Iteration    1    CC    50.902771    ERROR.y    -61.88684


                          Predicted Values

                                  y

                             -1.215784


                          Iteration Errors

                                  y

                             -61.88684

Observation    1    Iteration    2    CC     0.364806    ERROR.y    41.752112


                          Predicted Values

                                  y

                             -114.4503


                          Iteration Errors

                                  y

                              41.75211
```

Figure 20.75. SOLVE, ITPRINT Output

Numerical Integration

The differential equation system is numerically integrated to obtain a solution for the derivative variables at each data point. The integration is performed by evaluating the provided model at multiple points between each data point. The integration method used is a variable order, variable step-size backward difference scheme; for more detailed information, refer to Aiken (1985) and Byrne (1975). The step size or time step is chosen to satisfy a *local truncation error* requirement. The term *truncation error* comes from the fact that the integration scheme uses a truncated series expansion of the integrated function to do the integration. Because the series is truncated, the integration scheme is within the truncation error of the true value.

To further improve the accuracy of the integration, the total integration time is broken up into small intervals (time steps or step sizes), and the integration scheme is applied to those intervals. The integration at each time step uses the values computed at the previous time step so that the truncation error tends to accumulate. It is usually not possible to estimate the global error with much precision. The best that can be done is to monitor and to control the local truncation error, which is the truncation error committed at each time step relative to

$$d = \max_{0 \le t \le T} (\|y(t)\|_\infty, 1)$$

where $y(t)$ is the integrated variable. Furthermore, the $y(t)$s are dynamically scaled to within two orders of magnitude one to keep the error monitoring well behaved.

The local truncation error requirement defaults to $1.0E - 9$. You can specify the LTEBOUND= option to modify that requirement. The LTEBOUND= option is a relative measure of accuracy, so a value smaller than $1.0E - 10$ is usually not practical. A larger bound increases the speed of the simulation and estimation but decreases the accuracy of the results. If the LTEBOUND= option is set too small, the integrator is not able to take time steps small enough to satisfy the local truncation error requirement and still have enough machine precision to compute the results. Since the integrations are scaled to within $1.0E - 2$ of one, the simulated values should be correct to at least seven decimal places.

There is a default minimum time step of $1.0E - 14$. This minimum time step is controlled by the MINTIMESTEP= option and the machine epsilon. If the minimum time step is smaller than the machine epsilon times the final time value, the minimum time step is increased automatically.

For the points between each observation in the data set, the values for nonintegrated variables in the data set are obtained from a linear interpolation from the two closest points. Lagged variables can be used with integrations, but their values are discrete and are not interpolated between points. Lagging, therefore, can then be used to input step functions into the integration.

The derivatives necessary for estimation (the gradient with respect to the parameters) and goal seeking (the Jacobian) are computed by numerically integrating analytical derivatives. The accuracy of the derivatives is controlled by the same integration techniques mentioned previously.

Limitations

There are limitations to the types of differential equations that can be solved or estimated. One type is an explosive differential equation (finite escape velocity) for which the following differential equation is an example:

$$y^{'} = a{\times}y, \quad a > 0$$

If this differential equation is integrated too far in time, y exceeds the maximum value allowed on the computer, and the integration terminates.

Likewise, differential systems that are singular cannot be solved or estimated in general. For example, consider the following differential system:

$$\begin{aligned} x^{'} &= -y^{'} + 2x + 4y + \exp(\mathrm{t}) \\ y^{'} &= -x^{'} + y + \exp(4{*}\mathrm{t}) \end{aligned}$$

This system has an analytical solution, but an accurate numerical solution is very difficult to obtain. The reason is that $y^{'}$ and $x^{'}$ cannot be isolated on the left-hand side of the equation. If the equation is modified slightly to

$$\begin{aligned} x^{'} &= -y^{'} + 2x + 4y + \exp(\mathrm{t}) \\ y^{'} &= x^{'} + y + \exp(4\mathrm{t}) \end{aligned}$$

the system is nonsingular, but the integration process could still fail or be extremely slow. If the MODEL procedure encounters either system, a warning message is issued.

This system can be rewritten as the following recursive system,

$$\begin{aligned} x^{'} &= 0.5y + 0.5\exp(4t) + x + 1.5y - 0.5\exp(t) \\ y^{'} &= x^{'} + y + \exp(4\mathrm{t}) \end{aligned}$$

which can be estimated and simulated successfully with the MODEL procedure.

Petzold (1982) mentions a class of differential algebraic equations that, when integrated numerically, could produce incorrect or misleading results. An example of such a system mentioned in Petzold (1982) is

$$\begin{aligned} y_2^{'}(t) &= y_1(t) + g_1(t) \\ 0 &= y_2(t) + g_2(t) \end{aligned}$$

The analytical solution to this system depends on g and its derivatives at the current time only and not on its initial value or past history. You should avoid systems of this and other similar forms mentioned in Petzold (1982).

SOLVE Data Sets

SDATA= Input Data Set

The SDATA= option reads a cross-equation covariance matrix from a data set. The covariance matrix read from the SDATA= data set specified on the SOLVE statement is used to generate random equation errors when the RANDOM= option specifies Monte Carlo simulation.

Typically, the SDATA= data set is created by the OUTS= on a previous FIT statement. (The OUTS= data set from a FIT statement can be read back in by a SOLVE statement in the same PROC MODEL step.)

You can create an input SDATA= data set using the DATA step. PROC MODEL expects to find a character variable _NAME_ in the SDATA= data set as well as variables for the equations in the estimation or solution. For each observation with a _NAME_ value matching the name of an equation, PROC MODEL fills the corresponding row of the S matrix with the values of the names of equations found in the data set. If a row or column is omitted from the data set, an identity matrix row or column is assumed. Missing values are ignored. Since the S matrix is symmetric, you can include only a triangular part of the S matrix in the SDATA= data set with the omitted part indicated by missing values. If the SDATA= data set contains multiple observations with the same _NAME_, the last values supplied for the _NAME_ variable are used. The "OUTS= Data Set" section contains more details on the format of this data set.

Use the TYPE= option to specify the type of estimation method used to produce the S matrix you want to input.

ESTDATA= Input Data Set

The ESTDATA= option specifies an input data set that contains an observation with values for some or all of the model parameters. It can also contain observations with the rows of a covariance matrix for the parameters.

When the ESTDATA= option is used, parameter values are set from the first observation. If the RANDOM= option is used and the ESTDATA= data set contains a covariance matrix, the covariance matrix of the parameter estimates is read and used to generate pseudo-random shocks to the model parameters for Monte Carlo simulation. These random perturbations have a multivariate normal distribution with the covariance matrix read from the ESTDATA= data set.

The ESTDATA= data set is usually created by the OUTEST= option in a FIT statement. The OUTEST= data set contains the parameter estimates produced by the FIT statement and also contains the estimated covariance of the parameter estimates if the OUTCOV option is used. This OUTEST= data set can be read in by the ESTDATA= option in a SOLVE statement.

You can also create an ESTDATA= data set with a SAS DATA step program. The data set must contain a numeric variable for each parameter to be given a value or covariance column. The name of the variable in the ESTDATA= data set must match the name of the parameter in the model. Parameters with names longer than 32 characters cannot be set from an ESTDATA= data set. The data set must also contain a character variable _NAME_ of length 32. _NAME_ has a blank value for the observation that gives values to the parameters. _NAME_ contains the name of a parameter for observations defining rows of the covariance matrix.

More than one set of parameter estimates and covariances can be stored in the ESTDATA= data set if the observations for the different estimates are identified by the variable _TYPE_. _TYPE_ must be a character variable of length eight. The TYPE= option is used to select for input the part of the ESTDATA= data set for which the value of the _TYPE_ variable matches the value of the TYPE= option.

OUT= Data Set

The OUT= data set contains solution values, residual values, and actual values of the solution variables.

The OUT= data set contains the following variables:

- BY variables
- RANGE variable
- ID variables
- _TYPE_, a character variable of length eight identifying the type of observation. The _TYPE_ variable can be PREDICT, RESIDUAL, ACTUAL, or ERROR.
- _MODE_, a character variable of length eight identifying the solution mode. _MODE_ takes the value FORECAST or SIMULATE.
- if lags are used, a numeric variable, _LAG_, containing the number of dynamic lags that contribute to the solution. The value of _LAG_ is always zero for STATIC mode solutions. _LAG_ is set to a missing value for lag-starting observations.
- _REP_, a numeric variable containing the replication number, if the RANDOM= option is used. For example, if RANDOM=10, each input observation results in eleven output observations with _REP_ values 0 through 10. The observations with _REP_=0 are from the unperturbed solution. (The random-number generator functions are suppressed, and the parameter and endogenous perturbations are zero when _REP_=0.)
- _ERRORS_, a numeric variable containing the number of errors that occurred during the execution of the program for the last iteration for the observation. If the solution failed to converge, this is counted as one error, and the _ERRORS_ variable is made negative.
- solution and other variables. The solution variables contain solution or predicted values for _TYPE_=PREDICT observations, residuals for

TYPE=RESIDUAL observations, or actual values for _TYPE_=ACTUAL observations. The other model variables, and any other variables read from the input data set, are always actual values from the input data set.

- any other variables named in the OUTVARS statement. These can be program variables computed by the model program, CONTROL variables, parameters, or special variables in the model program. Compound variable names longer than 32 characters are truncated in the OUT= data set.

By default only the predicted values are written to the OUT= data set. The OUTRESID, OUTACTUAL, and OUTERROR options are used to add the residual, actual, and ERROR. values to the data set.

For examples of the OUT= data set, see Example 20.6 at the end of this chapter.

DATA= Input Data Set

The input data set should contain all of the exogenous variables and should supply nonmissing values for them for each period to be solved.

Solution variables can be supplied in the input data set and are used as follows:

- to supply initial lags. For example, if the lag length of the model is three, three observations are read in to feed the lags before any solutions are computed.
- to evaluate the goodness of fit. Goodness-of-fit measures are computed based on the difference between the solved values and the actual values supplied from the data set.
- to supply starting values for the iterative solution. If the value from the input data set for a solution variable is missing, the starting value for it is taken from the solution of the last period (if nonmissing) or else the solution estimate is started at zero.
- For STATIC mode solutions, actual values from the data set are used by the lagging functions for the solution variables.
- for FORECAST mode solutions, actual values from the data set are used as the solution values when nonmissing.

Programming Language Overview

Variables in the Model Program

Variable names are alphanumeric but must start with a letter. The length is limited to thirty-two characters.

PROC MODEL uses several classes of variables, and different variable classes are treated differently. Variable class is controlled by *declaration statements*. These are the VAR, ENDOGENOUS, and EXOGENOUS statements for model variables, the PARAMETERS statement for parameters, and the CONTROL statement for control class variables. These declaration statements have several valid abbreviations. Various *internal variables* are also made available to the model program to allow communication between the model program and the procedure. RANGE, ID, and BY variables are also available to the model program. Those variables not declared as any of the preceding classes are *program variables*.

Some classes of variables can be lagged; that is, their value at each observation is remembered, and previous values can be referred to by the lagging functions. Other classes have only a single value and are not affected by lagging functions. For example, parameters have only one value and are not affected by lagging functions; therefore, if P is a parameter, DIF*n*(P) is always 0, and LAG*n*(P) is always the same as P for all values of *n*.

The different variable classes and their roles in the model are described in the following.

Model Variables

Model variables are declared by VAR, ENDOGENOUS, or EXOGENOUS statements, or by FIT and SOLVE statements. The model variables are the variables that the model is intended to explain or predict.

PROC MODEL allows you to use expressions on the left-hand side of the equal sign to define model equations. For example, a log linear model for Y can be written as

```
log( y ) = a + b * x;
```

Previously, only a variable name was allowed on the left-hand side of the equal sign.

The text on the left hand side of the equation serves as the equation name used to identify the equation in printed output, in the OUT= data sets, and in FIT or SOLVE statements. To refer to equations specified using left-hand side expressions (on the FIT statement, for example), place the left-hand side expression in quotes. For example, the following statements fit a log linear model to the dependent variable Y:

```
proc model data=in;
   log( y ) = a + b * x;
   fit "log(y)";
run;
```

The estimation and simulation is performed by transforming the models into general form equations. No actual or predicted value is available for general form equations so no R^2 or adjusted R^2 will be computed.

Equation Variables

An equation variable is one of several special variables used by PROC MODEL to control the evaluation of model equations. An equation variable name consists of one of the prefixes EQ, RESID, ERROR, PRED, or ACTUAL, followed by a period and the name of a model equation.

Equation variable names can appear on parts of the PROC MODEL printed output, and they can be used in the model program. For example, RESID-prefixed variables can be used in LAG functions to define equations with moving-average error terms. See the "Autoregressive Moving-Average Error Processes" section earlier in this chapter for details.

The meaning of these prefixes is detailed in the "Equation Translations" section.

Parameters

Parameters are variables that have the same value for each observation. Parameters can be given values or can be estimated by fitting the model to data. During the SOLVE stage, parameters are treated as constants. If no estimation is performed, the SOLVE stage uses the initial value provided in either the ESTDATA= data set, the MODEL= file, or on the PARAMETER statement, as the value of the parameter.

The PARAMETERS statement declares the parameters of the model. Parameters are not lagged, and they cannot be changed by the model program.

Control Variables

Control variables supply constant values to the model program that can be used to control the model in various ways. The CONTROL statement declares control variables and specifies their values. A control variable is like a parameter except that it has a fixed value and is not estimated from the data.

Control variables are not reinitialized before each pass through the data and can thus be used to retain values between passes. You can use control variables to vary the program logic. Control variables are not affected by lagging functions.

For example, if you have two versions of an equation for a variable Y, you could put both versions in the model and, using a CONTROL statement to select one of them, produce two different solutions to explore the effect the choice of equation has on the model:

```
      select (case);
         when (1) y =  ...first version of equation... ;
         when (2) y =  ...second version of equation... ;
      end;
      control case 1;
      solve / out=case1;
   run;
```

```
      control case 2;
      solve / out=case2;
   run;
```

RANGE, ID, and BY Variables

The RANGE statement controls the range of observations in the input data set that is processed by PROC MODEL. The ID statement lists variables in the input data set that are used to identify observations on the printout and in the output data set. The BY statement can be used to make PROC MODEL perform a separate analysis for each BY group. The variable in the RANGE statement, the ID variables, and the BY variables are available for the model program to examine, but their values should not be changed by the program. The BY variables are not affected by lagging functions.

Internal Variables

You can use several internal variables in the model program to communicate with the procedure. For example, if you wanted PROC MODEL to list the values of all the variables when more than 10 iterations are performed and the procedure is past the 20th observation, you can write

```
   if _obs_ > 20 then if _iter_ > 10 then _list_ = 1;
```

Internal variables are not affected by lagging functions, and they cannot be changed by the model program except as noted. The following internal variables are available. The variables are all numeric except where noted.

ERRORS	a flag that is set to 0 at the start of program execution and is set to a nonzero value whenever an error occurs. The program can also set the _ERRORS_ variable.
ITER	the iteration number. For FIT tasks, the value of _ITER_ is negative for preliminary grid-search passes. The iterative phase of the estimation starts with iteration 0. After the estimates have converged, a final pass is made to collect statistics with _ITER_ set to a missing value. Note that at least one pass, and perhaps several subiteration passes as well, is made for each iteration. For SOLVE tasks, _ITER_ counts the iterations used to compute the simultaneous solution of the system.
LAG	the number of dynamic lags that contribute to the solution at the current observation. _LAG_ is always 0 for FIT tasks and for STATIC solutions. _LAG_ is set to a missing value during the lag starting phase.
LIST	list flag that is set to 0 at the start of program execution. The program can set _LIST_ to a nonzero value to request a listing of the values of all the variables in the program after the program has finished executing.

METHOD	is the solution method in use for SOLVE tasks. _METHOD_ is set to a blank value for FIT tasks. _METHOD_ is a character-valued variable. Values are NEWTON, JACOBI, SIEDEL, or ONEPASS.
MODE	takes the value ESTIMATE for FIT tasks and the value SIMULATE or FORECAST for SOLVE tasks. _MODE_ is a character-valued variable.
NMISS	the number of missing or otherwise unusable observations during the model estimation. For FIT tasks, _NMISS_ is initially set to 0; at the start of each iteration, _NMISS_ is set to the number of unusable observations for the previous iteration. For SOLVE tasks, _NMISS_ is set to a missing value.
NUSED	the number of nonmissing observations used in the estimation. For FIT tasks, PROC MODEL initially sets _NUSED_ to the number of parameters; at the start of each iteration, _NUSED_ is reset to the number of observations used in the previous iteration. For SOLVE tasks, _NUSED_ is set to a missing value.
OBS	counts the observations being processed. _OBS_ is negative or 0 for observations in the lag starting phase.
REP	the replication number for Monte Carlo simulation when the RANDOM= option is specified in the SOLVE statement. _REP_ is 0 when the RANDOM= option is not used and for FIT tasks. When _REP_=0, the random-number generator functions always return 0.
WEIGHT	the weight of the observation. For FIT tasks, _WEIGHT_ provides a weight for the observation in the estimation. _WEIGHT_ is initialized to 1.0 at the start of execution for FIT tasks. For SOLVE tasks, _WEIGHT_ is ignored.

Program Variables

Variables not in any of the other classes are called program variables. Program variables are used to hold intermediate results of calculations. Program variables are reinitialized to missing values before each observation is processed. Program variables can be lagged. The RETAIN statement can be used to give program variables initial values and enable them to keep their values between observations.

Character Variables

PROC MODEL supports both numeric and character variables. Character variables are not involved in the model specification but can be used to label observations, to write debugging messages, or for documentation purposes. All variables are numeric unless they are the following.

- character variables in a DATA= SAS data set
- program variables assigned a character value
- declared to be character by a LENGTH or ATTRIB statement.

Equation Translations

Equations written in normalized form are always automatically converted to general form equations. For example, when a normalized-form equation such as

```
y = a + b*x;
```

is encountered, it is translated into the equations

```
PRED.y = a + b*x;
RESID.y = PRED.y - ACTUAL.y;
ERROR.y = PRED.y - y;
```

If the same system is expressed as the following general-form equation, then this equation is used unchanged.

```
EQ.y = y -  a + b*x;
```

This makes it easy to solve for arbitrary variables and to modify the error terms for autoregressive or moving average models.

Use the LIST option to see how this transformation is performed. For example, the following statements produce the listing shown in Figure 20.76.

```
proc model data=line list;
    y = a1 + b1*x1 + c1*x2;
    fit y;
run;
```

```
                              The MODEL Procedure

                        Listing of Compiled Program Code
   Stmt     Line:Col       Statement as Parsed

      1     15820:39       PRED.y = a1 + b1 * x1 + c1 * x2;
      1     15820:39       RESID.y = PRED.y - ACTUAL.y;
      1     15820:39       ERROR.y = PRED.y - y;
```

Figure 20.76. LIST Output

PRED.Y is the predicted value of Y, and ACTUAL.Y is the value of Y in the data set. The predicted value minus the actual value, RESID.Y, is then the error term, ϵ, for the original Y equation. ACTUAL.Y and Y have the same value for parameter estimation. For solve tasks, ACTUAL.Y is still the value of Y in the data set but Y becomes the solved value; the value that satisfies PRED.Y - Y = 0.

The following are the equation variable definitions.

EQ.	The value of an EQ-prefixed equation variable (normally used to define a general-form equation) represents the failure of the equation to hold. When the EQ.*name* variable is 0, the *name* equation is satisfied.
RESID.	The RESID.*name* variables represent the stochastic parts of the equations and are used to define the objective function for the estimation process. A RESID.-prefixed equation variable is like an EQ-prefixed variable but makes it possible to use or transform the stochastic part of the equation. The RESID. equation is used in place of the ERROR. equation for model solutions if it has been reassigned or used in the equation.
ERROR.	An ERROR.*name* variable is like an EQ-prefixed variable, except that it is used only for model solution and does not affect parameter estimation.
PRED.	For a normalized-form equation (specified by assignment to a model variable), the PRED.*name* equation variable holds the predicted value, where *name* is the name of both the model variable and the corresponding equation. (PRED-prefixed variables are not created for general-form equations.)
ACTUAL.	For a normalized-form equation (specified by assignment to a model variable), the ACTUAL.*name* equation variable holds the value of the *name* model variable read from the input data set.
DERT.	The DERT.*name* variable defines a differential equation. Once defined, it may be used on the right-hand side of another equation.
H.	The H.*name* variable specifies the functional form for the variance of the named equation.
GMM_H.	This is created for H.*vars* and is the moment equation for the variance for GMM. This variable is used only for GMM.

```
GMM_H.name = RESID.name**2 - H.name;
```

MSE.	The MSE.*y* variable contains the value of the mean square error for *y* at each iteration. An MSE. variable is created for each dependent/endogenous variable in the model. These variables can be used to specify the missing lagged values in the estimation and simulation of GARCH type models.

```
demret = intercept ;
h.demret = arch0 +
             arch1 * xlag( resid.demret ** 2, mse.demret) +
             garch1 * zlag(h.demret, mse.demret) ;
```

NRESID.	This is created for H.*vars* and is the normalized residual of the variable <*name*>. The formula is

```
NRESID.name = RESID.name/ sqrt(H.name);
```

The three equation variable prefixes, RESID., ERROR., and EQ. allow for control over the objective function for the FIT, the SOLVE, or both the FIT and the SOLVE stages. For FIT tasks, PROC MODEL looks first for a RESID.*name* variable for each equation. If defined, the RESID-prefixed equation variable is used to define the objective function for the parameter estimation process. Otherwise, PROC MODEL looks for an EQ-prefixed variable for the equation and uses it instead.

For SOLVE tasks, PROC MODEL looks first for an ERROR.*name* variable for each equation. If defined, the ERROR-prefixed equation variable is used for the solution process. Otherwise, PROC MODEL looks for an EQ-prefixed variable for the equation and uses it instead. To solve the simultaneous equation system, PROC MODEL computes values of the solution variables (the model variables being solved for) that make all of the ERROR.name and EQ.*name* variables close to 0.

Derivatives

Nonlinear modeling techniques require the calculation of derivatives of certain variables with respect to other variables. The MODEL procedure includes an analytic differentiator that determines the model derivatives and generates program code to compute these derivatives. When parameters are estimated, the MODEL procedure takes the derivatives of the equation with respect to the parameters. When the model is solved, Newton's method requires the derivatives of the equations with respect to the variables solved for.

PROC MODEL uses exact mathematical formulas for derivatives of non-user-defined functions. For other functions, numerical derivatives are computed and used.

The differentiator differentiates the entire model program, including conditional logic and flow of control statements. Delayed definitions, as when the LAG of a program variable is referred to before the variable is assigned a value, are also differentiated correctly.

The differentiator includes optimization features that produce efficient code for the calculation of derivatives. However, when flow of control statements such as GOTO statements are used, the optimization process is impeded, and less efficient code for derivatives may be produced. Optimization is also reduced by conditional statements, iterative DO loops, and multiple assignments to the same variable.

The table of derivatives is printed with the LISTDER option. The code generated for the computation of the derivatives is printed with the LISTCODE option.

Derivative Variables

When the differentiator needs to generate code to evaluate the expression for the derivative of a variable, the result is stored in a special derivative variable. Derivative variables are not created when the derivative expression reduces to a previously computed result, a variable, or a constant. The names of derivative variables, which may sometimes appear in the printed output, have the form *@obj*/*@wrt*, where *obj* is the variable whose derivative is being taken and *wrt* is the variable that the differentiation is with respect to. For example, the derivative variable for the derivative of *Y* with respect to *X* is named *@Y/@X*.

The derivative variables cannot be accessed or used as part of the model program.

Mathematical Functions

The following is a brief summary of SAS functions useful for defining models. Additional functions and details are in *SAS Language: Reference*. Information on creating new functions can be found in *SAS/TOOLKIT Software: Usage and Reference*, chapter 15, "Writing a SAS Function or Call Routine."

ABS(*x*)	the absolute value of *x*
ARCOS(*x*)	the arccosine in radians of *x*. *x* should be between -1 and 1.
ARSIN(*x*)	the arcsine in radians of *x*. *x* should be between -1 and 1.
ATAN(*x*)	the arctangent in radians of *x*
COS(*x*)	the cosine of *x*. *x* is in radians.
COSH(*x*)	the hyperbolic cosine of *x*
EXP(*x*)	e^{x}
LOG(*x*)	the natural logarithm of *x*
LOG10(*x*)	the log base ten of *x*
LOG2(*x*)	the log base two of *x*
SIN(*x*)	the sine of *x*. *x* is in radians.
SINH(*x*)	the hyperbolic sine of *x*
SQRT(*x*)	the square root of *x*
TAN(*x*)	the tangent of *x*. *x* is in radians and is not an odd multiple of $\pi/2$.
TANH(*x*)	the hyperbolic tangent of *x*

Random-Number Functions

The MODEL procedure provides several functions for generating random numbers for Monte Carlo simulation. These functions use the same generators as the corresponding SAS DATA step functions.

The following random-number functions are supported: RANBIN, RANCAU, RAND, RANEXP, RANGAM, RANNOR, RANPOI, RANTBL, RANTRI, and RANUNI. For more information, refer to *SAS Language: Reference*.

Each reference to a random-number function sets up a separate pseudo-random sequence. Note that this means that two calls to the same random function with the same seed produce identical results. This is different from the behavior of the random-number functions used in the SAS DATA step. For example, the statements

```
x=rannor(123);
y=rannor(123);
z=rannor(567);
q=rand('BETA', 1, 12 );
```

produce identical values for X and Y, but Z is from an independent pseudo-random sequence.

For FIT tasks, all random-number functions always return 0. For SOLVE tasks, when Monte Carlo simulation is requested, a random-number function computes a new random number on the first iteration for an observation (if it is executed on that iteration) and returns that same value for all later iterations of that observation. When Monte Carlo simulation is not requested, random-number functions always return 0.

Functions Across Time

PROC MODEL provides four types of special built-in functions that refer to the values of variables and expressions in previous time periods. These functions have the form

LAG*n*(**[** *i* **,]** *x*)	returns the *i*th lag of *x*, where *n* is the maximum lag;
DIF*n*(*x*)	difference of *x* at lag *n*
ZLAG*n*(**[** *i* **,]** *x*)	returns the *i*th lag of *x*, where *n* is the maximum lag, with missing lags replaced with zero;
XLAG*n*(*x* **,** *y*)	returns the *n*th lag of *x* if *x* is nonmissing, or *y* if x is missing;
ZDIF*n*(*x*)	difference with lag length truncated and missing values converted to zero;
MOVAVGn(*x*)	the width of the moving average is *n*, and *x* is the variable or expression to compute the moving average of. Missing values of *x* are omitted in computing the average.

where *n* represents the number of periods, and *x* is any expression. The argument *i* is a variable or expression giving the lag length ($0 <= i <= n$), if the index value *i* is omitted, the maximum lag length *n* is used.

If you do not specify *n*, the number of periods is assumed to be one. For example, LAG(X) is the same as LAG1(X). No more than four digits can be used with a lagging function; that is, LAG9999 is the greatest LAG function, ZDIF9999 is the greatest ZDIF function, and so on.

The LAG functions get values from previous observations and make them available to the program. For example, LAG(X) returns the value of the variable X as it was computed in the execution of the program for the preceding observation. The expression LAG2(X+2*Y) returns the value of the expression X+2*Y, computed using the values of the variables X and Y that were computed by the execution of the program for the observation two periods ago.

The DIF functions return the difference between the current value of a variable or expression and the value of its LAG. For example, DIF2(X) is a short way of writing X-LAG2(X), and DIF15(SQRT(2*Z)) is a short way of writing SQRT(2*Z)-LAG15(SQRT(2*Z)).

The ZLAG and ZDIF functions are like the LAG and DIF functions, but they are not counted in the determination of the program lag length, and they replace missing values with 0s. The ZLAG function returns the lagged value if the lagged value is nonmissing, or 0 if the lagged value is missing. The ZDIF function returns the differenced value if the differenced value is nonmissing, or 0 if the value of the differenced value is missing. The ZLAG function is especially useful for models with ARMA error processes. See "Lag Logic", which follows for details.

Lag Logic

The LAG and DIF lagging functions in the MODEL procedure are different from the queuing functions with the same names in the DATA step. Lags are determined by the final values that are set for the program variables by the execution of the model program for the observation. This can have upsetting consequences for programs that take lags of program variables that are given different values at various places in the program, for example,

```
   temp = x + w;
   t    = lag( temp );
   temp = q - r;
   s    = lag( temp );
```

The expression LAG(TEMP) always refers to LAG(Q-R), never to LAG(X+W), since Q-R is the final value assigned to the variable TEMP by the model program. If LAG(X+W) is wanted for T, it should be computed as T=LAG(X+W) and not T=LAG(TEMP), as in the preceding example.

Care should also be exercised in using the DIF functions with program variables that may be reassigned later in the program. For example, the program

```
   temp =  x ;
   s    = dif( temp );
   temp = 3 * y;
```

computes values for S equivalent to

```
   s =  x  - lag( 3 * y );
```

Note that in the preceding examples, TEMP is a program variable, *not* a model variable. If it were a model variable, the assignments to it would be changed to assignments to a corresponding equation variable.

Note that whereas LAG1(LAG1(X)) is the same as LAG2(X), DIF1(DIF1(X)) is *not* the same as DIF2(X). The DIF2 function is the difference between the current period value at the point in the program where the function is executed and the final value at the end of execution two periods ago; DIF2 is not the second difference. In contrast, DIF1(DIF1(X)) is equal to DIF1(X)-LAG1(DIF1(X)), which equals X-2*LAG1(X)+LAG2(X), which is the second difference of X.

More information on the differences between PROC MODEL and the DATA step LAG and DIF functions is found in Chapter 2, “Working with Time Series Data.” .

Lag Lengths

The lag length of the model program is the number of lags needed for any relevant equation. The program lag length controls the number of observations used to initialize the lags.

PROC MODEL keeps track of the use of lags in the model program and automatically determines the lag length of each equation and of the model as a whole. PROC MODEL sets the program lag length to the maximum number of lags needed to compute any equation to be estimated, solved, or needed to compute any instrument variable used.

In determining the lag length, the ZLAG and ZDIF functions are treated as always having a lag length of 0. For example, if Y is computed as

```
   y = lag2( x + zdif3( temp ) );
```

then Y has a lag length of 2 (regardless of how TEMP is defined). If Y is computed as

```
   y = zlag2( x + dif3( temp ) );
```

then Y has a lag length of 0.

This is so that ARMA errors can be specified without causing the loss of additional observations to the lag starting phase and so that recursive lag specifications, such as moving-average error terms, can be used. Recursive lags are not permitted unless the ZLAG or ZDIF functions are used to truncate the lag length. For example, the following statement produces an error message:

```
   t = a + b * lag( t );
```

The program variable T depends recursively on its own lag, and the lag length of T is therefore undefined.

In the following equation RESID.Y depends on the predicted value for the Y equation but the predicted value for the Y equation depends on the LAG of RESID.Y, and, thus, the predicted value for the Y equation depends recursively on its own lag.

```
   y = yhat + ma * lag( resid.y );
```

The lag length is infinite, and PROC MODEL prints an error message and stops. Since this kind of specification is allowed, the recursion must be truncated at some point. The ZLAG and ZDIF functions do this.

The following equation is legal and results in a lag length for the Y equation equal to the lag length of YHAT:

```
   y = yhat + ma * zlag( resid.y );
```

Initially, the lags of RESID.Y are missing, and the ZLAG function replaces the missing residuals with 0s, their unconditional expected values.

The ZLAG0 function can be used to zero out the lag length of an expression. ZLAG0(*x*) returns the current period value of the expression *x*, if nonmissing, or else returns 0, and prevents the lag length of *x* from contributing to the lag length of the current statement.

Initializing Lags

At the start of each pass through the data set or BY group, the lag variables are set to missing values and an initialization is performed to fill the lags. During this phase, observations are read from the data set, and the model variables are given values from the data. If necessary, the model is executed to assign values to program variables that are used in lagging functions. The results for variables used in lag functions are saved. These observations are not included in the estimation or solution.

If, during the execution of the program for the lag starting phase, a lag function refers to lags that are missing, the lag function returns missing. Execution errors that occur while starting the lags are not reported unless requested. The modeling system automatically determines whether the program needs to be executed during the lag starting phase.

If L is the maximum lag length of any equation being fit or solved, then the first L observations are used to prime the lags. If a BY statement is used, the first L observations in the BY group are used to prime the lags. If a RANGE statement is used, the first L observations prior to the first observation requested in the RANGE statement are used to prime the lags. Therefore, there should be at least L observations in the data set.

Initial values for the lags of model variables can also be supplied in VAR, ENDOGENOUS, and EXOGENOUS statements. This feature provides initial lags of solution variables for dynamic solution when initial values for the solution variable are not available in the input data set. For example, the statement

```
   var x 2 3 y 4 5 z 1;
```

feeds the initial lags exactly like these values in an input data set:

Lag	X	Y	Z
2	3	5	.
1	2	4	1

If initial values for lags are available in the input data set and initial lag values are also given in a declaration statement, the values in the VAR, ENDOGENOUS, or EXOGENOUS statements take priority.

The RANGE statement is used to control the range of observations in the input data set that are processed by PROC MODEL. In the statement

```
   range date = '01jan1924'd to '01dec1943'd;
```

'01jan1924' specifies the starting period of the range, and '01dec1943' specifies the ending period. The observations in the data set immediately prior to the start of the range are used to initialize the lags.

Language Differences

For the most part, PROC MODEL programming statements work the same as they do in the DATA step as documented in *SAS Language: Reference*. However, there are several differences that should be noted.

DO Statement Differences

The DO statement in PROC MODEL does not allow a character index variable. Thus, the following DO statement is not valid in PROC MODEL, although it is supported in the DATA step:

```
   do i = 'A', 'B', 'C';         /* invalid PROC MODEL code */
```

IF Statement Differences

The IF statement in PROC MODEL does not allow a character-valued condition. For example, the following IF statement is not supported by PROC MODEL:

```
   if 'this' then  statement;
```

Comparisons of character values are supported in IF statements, so the following IF statement is acceptable:

```
   if 'this' < 'that' then  statement};
```

PROC MODEL allows for embedded conditionals in expressions. For example the following two statements are equivalent:

```
   flag = if time = 1 or time = 2 then conc+30/5 + dose*time
                else if time > 5 then (0=1) else (patient * flag);

   if time = 1 or time = 2 then flag= conc+30/5 + dose*time;
          else if time > 5 then flag=(0=1); else flag=patient*flag;
```

Note that the ELSE operator only involves the first object or token after it so that the following assignments are not equivalent:

```
   total = if sum > 0 then sum else sum + reserve;
   total = if sum > 0 then sum else (sum + reserve);
```

The first assignment makes TOTAL always equal to SUM plus RESERVE.

PUT Statement Differences

The PUT statement, mostly used in PROC MODEL for program debugging, only supports some of the features of the DATA step PUT statement. It also has some new features that the DATA step PUT statement does not support.

The PROC MODEL PUT statement does not support line pointers, factored lists, iteration factors, overprinting, the _INFILE_ option, or the colon (:) format modifier.

The PROC MODEL PUT statement does support expressions but an expression must be enclosed in parentheses. For example, the following statement prints the square root of x:

```
   put (sqrt(x));
```

Subscripted array names must be enclosed in parentheses. For example, the following statement prints the *i*th element of the array A:

```
   put (a i);
```

However, the following statement is an error:

```
   put a i;
```

The PROC MODEL PUT statement supports the print item _PDV_ to print a formatted listing of all the variables in the program. For example, the following statement prints a much more readable listing of the variables than does the _ALL_ print item:

```
   put _pdv_;
```

To print all the elements of the array A, use the following statement:

```
   put a;
```

To print all the elements of A with each value labeled by the name of the element variable, use the statement

```
   put a=;
```

ABORT Statement Difference

In the MODEL procedure, the ABORT statement does not allow any arguments.

SELECT/WHEN/OTHERWISE Statement Differences

The WHEN and OTHERWISE statements allow more than one target statement. That is, DO groups are not necessary for multiple statement WHENs. For example in PROC MODEL, the following syntax is valid:

```
select;
   when(exp1)
      stmt1;
      stmt2;
   when(exp2)
      stmt3;
      stmt4;
end;
```

The ARRAY Statement

ARRAY *arrayname [{dimensions}] [$ [length]] [variables and constants];*

The ARRAY statement is used to associate a name with a list of variables and constants. The array name can then be used with subscripts in the model program to refer to the items in the list.

In PROC MODEL, the ARRAY statement does not support all the features of the DATA step ARRAY statement. Implicit indexing cannot be used; all array references must have explicit subscript expressions. Only exact array dimensions are allowed; lower-bound specifications are not supported. A maximum of six dimensions is allowed.

On the other hand, the ARRAY statement supported by PROC MODEL does allow both variables and constants to be used as array elements. You cannot make assignments to constant array elements. Both dimension specification and the list of elements are optional, but at least one must be supplied. When the list of elements is not given or fewer elements than the size of the array are listed, array variables are created by suffixing element numbers to the array name to complete the element list.

The following are valid PROC MODEL array statements:

```
array x[120];           /* array X of length 120           */
array q[2,2];           /* Two dimensional array Q         */
array b[4] va vb vc vd; /* B[2] = VB, B[4] = VD            */
array x x1-x30;         /* array X of length 30, X[7] = X7  */
array a[5] (1 2 3 4 5); /* array A initialized to 1,2,3,4,5 */
```

RETAIN Statement

RETAIN *variables initial-values ;*

The RETAIN statement causes a program variable to hold its value from a previous observation until the variable is reassigned. The RETAIN statement can be used to initialize program variables.

The RETAIN statement does not work for model variables, parameters, or control variables because the values of these variables are under the control of PROC MODEL and not programming statements. Use the PARMS and CONTROL statements to initialize parameters and control variables. Use the VAR, ENDOGENOUS, or EXOGENOUS statement to initialize model variables.

Storing Programs in Model Files

Models can be saved and recalled from SAS catalog files. SAS catalogs are special files that can store many kinds of data structures as separate units in one SAS file. Each separate unit is called an entry, and each entry has an entry type that identifies its structure to the SAS system.

In general, to save a model, use the OUTMODEL=*name* option on the PROC MODEL statement, where *name* is specified as *libref.catalog.entry*, *libref.entry*, or *entry*. The *libref*, *catalog*, and *entry* names must be valid SAS names no more than 32 characters long. The *catalog* name is restricted to seven characters on the CMS operating system. If not given, the *catalog* name defaults to MODELS, and the *libref* defaults to WORK. The entry type is always MODEL. Thus, OUTMODEL=X writes the model to the file WORK.MODELS.X.MODEL.

The MODEL= option is used to read in a model. A list of model files can be specified in the MODEL= option, and a range of names with numeric suffixes can be given, as in MODEL=(MODEL1-MODEL10). When more than one model file is given, the list must be placed in parentheses, as in MODEL=(A B C), except in the case of a single name. If more than one model file is specified, the files are combined in the order listed in the MODEL= option.

When the MODEL= option is specified in the PROC MODEL statement and model definition statements are also given later in the PROC MODEL step, the model files are read in first, in the order listed, and the model program specified in the PROC MODEL step is appended after the model program read from the MODEL= files. The class assigned to a variable, when multiple model files are used, is the last declaration of that variable. For example, if Y1 was declared endogenous in the model file M1 and exogenous in the model file M2, the following statement will cause Y1 to be declared exogenous.

```
   proc model model=(m1 m2);
```

The INCLUDE statement can be used to append model code to the current model code. In contrast, when the MODEL= option is used on the RESET statement, the current model is deleted before the new model is read.

No model file is output by default if the PROC MODEL step performs any FIT or SOLVE tasks, or if the MODEL= option or the NOSTORE option is used. However, to ensure compatibility with previous versions of SAS/ETS software, when the PROC MODEL step does nothing but compile the model program, no input model file is read, and the NOSTORE option is not used, a model file is written. This model file is the default input file for a later PROC SYSNLIN or PROC SIMNLIN step. The default output model filename in this case is WORK.MODELS._MODEL_.MODEL.

If FIT statements are used to estimate model parameters, the parameter estimates written to the output model file are the estimates from the last estimation performed for each parameter.

Diagnostics and Debugging

PROC MODEL provides several features to aid in finding errors in the model program. These debugging features are not usually needed; most models can be developed without them.

The example model program that follows will be used in the following sections to illustrate the diagnostic and debugging capabilities. This example is the estimation of a segmented model.

```
*---------Fitting a Segmented Model using MODEL----*
|      |                                           |
|   y  | quadratic          plateau                |
|      | y=a+b*x+c*x*x      y=p                    |
|      |                     .....................  |
|      |             .      :                      |
|      |         .          :                      |
|      |      .             :                      |
|      |    .               :                      |
|      |   .                :                      |
|      +-----------------------------------------X |
|                          x0                      |
|                                                  |
| continuity restriction: p=a+b*x0+c*x0**2         |
| smoothness restriction: 0=b+2*c*x0 so x0=-b/(2*c)|
*--------------------------------------------------*;
title 'QUADRATIC MODEL WITH PLATEAU';
data a;
   input y x @@;
   datalines;
.46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
.78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
.80 15 .78 16
;
proc model data=a;
parms a 0.45 b 0.5 c -0.0025;

x0 = -.5*b / c;      /* join point */
if x < x0 then       /* Quadratic part of model */
   y = a + b*x + c*x*x;
else                 /* Plateau part of model */
   y = a + b*x0 + c*x0*x0;

fit y;
run;
```

Program Listing

The LIST option produces a listing of the model program. The statements are printed one per line with the original line number and column position of the statement.

The program listing from the example program is shown in Figure 20.77.

```
                              QUADRATIC MODEL WITH PLATEAU

                                  The MODEL Procedure

                            Listing of Compiled Program Code
          Stmt    Line:Col      Statement as Parsed

             1    15888:74      x0 = (-0.5 * b) / c;
             2    15888:96      if x < x0 then
             3    15888:124     PRED.y = a + b * x + c * x * x;
             3    15888:124     RESID.y = PRED.y - ACTUAL.y;
             3    15888:124     ERROR.y = PRED.y - y;
             4    15888:148     else
             5    15888:176     PRED.y = a + b * x0 + c * x0 * x0;
             5    15888:176     RESID.y = PRED.y - ACTUAL.y;
             5    15888:176     ERROR.y = PRED.y - y;
```

Figure 20.77. LIST Output for Segmented Model

The LIST option also shows the model translations that PROC MODEL performs. LIST output is useful for understanding the code generated by the %AR and the %MA macros.

Cross-Reference

The XREF option produces a cross-reference listing of the variables in the model program. The XREF listing is usually used in conjunction with the LIST option. The XREF listing does not include derivative (@-prefixed) variables. The XREF listing does not include generated assignments to equation variables, PRED, RESID, and ERROR-prefixed variables, unless the DETAILS option is used.

The cross-reference from the example program is shown in Figure 20.78.

```
                                  The MODEL Procedure

                          Cross Reference Listing For Program
Symbol-----------        Kind     Type     References (statement)/(line):(col)

a                        Var      Num      Used: 3/15913:130 5/15913:182
b                        Var      Num      Used: 1/15913:82 3/15913:133 5/15913:185
c                        Var      Num      Used: 1/15913:85 3/15913:139 5/15913:192
x0                       Var      Num      Assigned: 1/15913:85
                                           Used: 2/15913:103 5/15913:185
                                           5/15913:192 5/15913:195
x                        Var      Num      Used: 2/15913:103 3/15913:133
                                           3/15913:139 3/15913:141
PRED.y                   Var      Num      Assigned: 3/15913:136 5/15913:189
```

Figure 20.78. XREF Output for Segmented Model

Compiler Listing

The LISTCODE option lists the model code and derivatives tables produced by the compiler. This listing is useful only for debugging and should not normally be needed.

LISTCODE prints the operator and operands of each operation generated by the compiler for each model program statement. Many of the operands are temporary variables generated by the compiler and given names such as #temp1. When derivatives are taken, the code listing includes the operations generated for the derivatives calculations. The derivatives tables are also listed.

A LISTCODE option prints the transformed equations from the example shown in Figure 20.79 and Figure 20.80.

```
                              The MODEL Procedure

                        Listing of Compiled Program Code
         Stmt    Line:Col      Statement as Parsed

            1    16459:83      x0 = (-0.5 * b) / c;
            1    16459:83      @x0/@b = -0.5 / c;
            1    16459:83      @x0/@c = (0 - x0) / c;
            2    16459:105     if x < x0 then
            3    16459:133     PRED.y = a + b * x + c * x * x;
            3    16459:133     @PRED.y/@a = 1;
            3    16459:133     @PRED.y/@b = x;
            3    16459:133     @PRED.y/@c = x * x;
            3    16459:133     RESID.y = PRED.y - ACTUAL.y;
            3    16459:133     @RESID.y/@a = @PRED.y/@a;
            3    16459:133     @RESID.y/@b = @PRED.y/@b;
            3    16459:133     @RESID.y/@c = @PRED.y/@c;
            3    16459:133     ERROR.y = PRED.y - y;
            4    16459:157     else
            5    16459:185     PRED.y = a + b * x0 + c * x0 * x0;
            5    16459:185     @PRED.y/@a = 1;
            5    16459:185     @PRED.y/@b = x0 + b * @x0/@b + (c
                               * @x0/@b * x0 + c * x0 * @x0/@b);
            5    16459:185     @PRED.y/@c = b * @x0/@c + ((x0 + c
                               * @x0/@c) * x0 + c * x0 * @x0/@c);
            5    16459:185     RESID.y = PRED.y - ACTUAL.y;
            5    16459:185     @RESID.y/@a = @PRED.y/@a;
            5    16459:185     @RESID.y/@b = @PRED.y/@b;
            5    16459:185     @RESID.y/@c = @PRED.y/@c;
            5    16459:185     ERROR.y = PRED.y - y;
```

Figure 20.79. LISTCODE Output for Segmented Model - Statements as Parsed

```
                              The MODEL Procedure


    1 Stmt ASSIGN      line 5619 column
                       83. (1) arg=x0
                       argsave=x0
                       Source Text:          x0 = -.5*b / c;
        Oper *         at 5619:91 (30,0,2).  * : #temp1 <- -0.5 b
        Oper /         at 5619:94 (31,0,2).  / : x0 <- #temp1 c
        Oper eeocf     at 5619:94 (18,0,1).  eeocf : _DER_ <- _DER_
        Oper /         at 5619:94 (31,0,2).  / : @x0/@b <- -0.5 c
        Oper -         at 5619:94 (33,0,2).  - : @1dt1_2 <- 0 x0
        Oper /         at 5619:94 (31,0,2).  / : @x0/@c <- @1dt1_2 c

    2 Stmt IF          line 5619 column       ref.st=ASSIGN stmt
                       105. (2) arg=#temp1   number 5 at 5619:185
                       argsave=#temp1
                       Source Text:          if x < x0 then
        Oper <         at 5619:112           < : #temp1 <- x x0
                       (36,0,2).

    3 Stmt ASSIGN      line 5619 column
                       133. (1) arg=PRED.y
                       argsave=y
                       Source Text:          y = a + b*x + c*x*x;
        Oper *         at 5619:142           * : #temp1 <- b x
                       (30,0,2).
        Oper +         at 5619:139           + : #temp2 <- a #temp1
                       (32,0,2).
        Oper *         at 5619:148           * : #temp3 <- c x
                       (30,0,2).
        Oper *         at 5619:150           * : #temp4 <- #temp3 x
                       (30,0,2).
        Oper +         at 5619:145           + : PRED.y <- #temp2 #temp4
                       (32,0,2).
        Oper eeocf     at 5619:150           eeocf : _DER_ <- _DER_
                       (18,0,1).
        Oper *         at 5619:150           * : @1dt1_1 <- x x
                       (30,0,2).
        Oper =         at 5619:145 (1,0,1).  = : @PRED.y/@a <- 1
        Oper =         at 5619:145 (1,0,1).  = : @PRED.y/@b <- x
        Oper =         at 5619:145 (1,0,1).  = : @PRED.y/@c <- @1dt1_1

    3 Stmt Assign      line 5619 column
                       133. (1) arg=RESID.y
                       argsave=y
        Oper -         at 5619:133           - : RESID.y <- PRED.y ACTUAL.y
                       (33,0,2).
        Oper eeocf     at 5619:133           eeocf : _DER_ <- _DER_
                       (18,0,1).
        Oper =         at 5619:133 (1,0,1).  = : @RESID.y/@a <- @PRED.y/@a
        Oper =         at 5619:133 (1,0,1).  = : @RESID.y/@b <- @PRED.y/@b
        Oper =         at 5619:133 (1,0,1).  = : @RESID.y/@c <- @PRED.y/@c

    3 Stmt Assign      line 5619 column
                       133. (1) arg=ERROR.y
                       argsave=y
        Oper -         at 5619:133           - : ERROR.y <- PRED.y y
                       (33,0,2).

    4 Stmt ELSE        line 5619 column
                       157. (9)
                       Source Text:          else
```

Figure 20.80. LISTCODE Output for Segmented Model - Compiled Code

Analyzing the Structure of Large Models

PROC MODEL provides several features to aid in analyzing the structure of the model program. These features summarize properties of the model in various forms.

The following Klein's model program is used to introduce the LISTDEP, BLOCK, and GRAPH options.

```
proc model  out=m data=klein listdep graph block;
   endogenous c p w i x wsum k y;
   exogenous  wp g t year;
   parms c0-c3 i0-i3 w0-w3;
   a: c = c0 + c1 * p + c2 * lag(p) + c3 * wsum;
   b: i = i0 + i1 * p + i2 * lag(p) + i3 * lag(k);
   c: w = w0 + w1 * x + w2 * lag(x) + w3 * year;
   x = c + i + g;
   y = c + i + g-t;
   p = x-w-t;
   k = lag(k) + i;
   wsum = w + wp;
   id year;
run;
```

Dependency List

The LISTDEP option produces a dependency list for each variable in the model program. For each variable, a list of variables that depend on it and a list of variables it depends on is given. The dependency list produced by the example program is shown in Figure 20.81.

```
                              The MODEL Procedure

                        Dependency Listing For Program
      Symbol-----------    Dependencies

      c                    Current values affect: ERROR.c PRED.x
                           RESID.x ERROR.x PRED.y RESID.y ERROR.y
      p                    Current values affect: PRED.c RESID.c
                           ERROR.c PRED.i RESID.i ERROR.i ERROR.p
                           Lagged values affect: PRED.c PRED.i
      w                    Current values affect: ERROR.w
                           PRED.p RESID.p ERROR.p PRED.wsum
                           RESID.wsum ERROR.wsum
      i                    Current values affect: ERROR.i PRED.x
                           RESID.x ERROR.x PRED.y RESID.y
                           ERROR.y PRED.k RESID.k ERROR.k
      x                    Current values affect: PRED.w RESID.w
                           ERROR.w ERROR.x PRED.p RESID.p ERROR.p
                           Lagged values affect: PRED.w
      wsum                 Current values affect: PRED.c
                           RESID.c ERROR.c ERROR.wsum
      k                    Current values affect: ERROR.k
                           Lagged values affect: PRED.i RESID.i
                           ERROR.i PRED.k RESID.k
```

Figure 20.81. A Portion of the LISTDEP Output for Klein's Model

BLOCK Listing

The BLOCK option prints an analysis of the program variables based on the assignments in the model program. The output produced by the example is shown in Figure 20.82.

```
                              The MODEL Procedure
                            Model Structure Analysis
              (Based on Assignments to Endogenous Model Variables)

                  Exogenous Variables     wp g t year
                  Endogenous Variables    c p w i x wsum k y
NOTE: The System Consists of 2 Recursive Equations and 1 Simultaneous Blocks.


                          Block Structure of the System

                           Block 1    c p w i x wsum


                       Dependency Structure of the System

                  Block 1    Depends On All_Exogenous
                  k          Depends On Block 1 All_Exogenous
                  y          Depends On Block 1 All_Exogenous
```

Figure 20.82. The BLOCK Output for Klein's Model

One use for the block output is to put a model in recursive form. Simulations of the model can be done with the SEIDEL method, which is efficient if the model is recursive and if the equations are in recursive order. By examining the block output, you can determine how to reorder the model equations for the most efficient simulation.

Adjacency Graph

The GRAPH option displays the same information as the BLOCK option with the addition of an adjacency graph. An X in a column in an adjacency graph indicates that the variable associated with the row depends on the variable associated with the column. The output produced by the example is shown in Figure 20.83.

The first and last graphs are straightforward. The middle graph represents the dependencies of the nonexogenous variables after transitive closure has been performed (that is, A depends on B, and B depends on C, so A depends on C). The preceding transitive closure matrix indicates that K and Y do not directly or indirectly depend on each other.

```
               The MODEL Procedure

      Adjacency Matrix for Graph of System
                                w           y
                                s           e
                                u     w     a
    Variable          c p w i x m k y p g t r

                                      * * * *
    c                 X X . . . X . . . . . .
    p                 . X X . X . . . . . X .
    w                 . . X . X . . . . . . X
    i                 . X . X . . . . . . . .
    x                 X . . X X . . . . X . .
    wsum              . . X . . X . . X . . .
    k                 . . . X . . X . . . . .
    y                 X . . X . . . X . X X .
    wp              * . . . . . . . . X . . .
    g               * . . . . . . . . . X . .
    t               * . . . . . . . . . . X .
    year            * . . . . . . . . . . . X

         (Note: * = Exogenous Variable.)


    Transitive Closure Matrix of Sorted System
                                         w
                                         s
                                         u
    Block    Variable          c p w i x m k y

        1    c                 X X X X X X . .
        1    p                 X X X X X X . .
        1    w                 X X X X X X . .
        1    i                 X X X X X X . .
        1    x                 X X X X X X . .
        1    wsum              X X X X X X . .
             k                 X X X X X X X .
             y                 X X X X X X . X


       Adjacency Matrix for Graph of System
             Including Lagged Impacts
                                     w           y
                                     s           e
                                     u     w     a
Block    Variable          c p w i x m k y p g t r

                                           * * * *
    1    c                 X L . . . X . . . . . .
    1    p                 . X X . X . . . . . X .
    1    w                 . . X . L . . . . . . X
    1    i                 . L . X . . L . . . . .
    1    x                 X . . X X . . . . X . .
    1    wsum              . . X . . X . . X . . .
         k                 . . . X . . L . . . . .
         y                 X . . X . . . X . X X .
         wp              * . . . . . . . . X . . .
         g               * . . . . . . . . . X . .
         t               * . . . . . . . . . . X .
         year            * . . . . . . . . . . . X

         (Note: * = Exogenous Variable.)
```

Figure 20.83. The GRAPH Output for Klein’s Model

Examples

Example 20.1. OLS Single Nonlinear Equation

This example illustrates the use of the MODEL procedure for nonlinear ordinary least-squares (OLS) regression. The model is a logistic growth curve for the population of the United States. The data is the population in millions recorded at ten year intervals starting in 1790 and ending in 2000. For an explanation of the starting values given by the START= option, see "Troubleshooting Convergence Problems" earlier in this chapter. Portions of the output from the following code are shown in Output 20.1.1 and Output 20.1.2.

```
title 'Logistic Growth Curve Model of U.S. Population';
data uspop;
   input pop :6.3 @@;
   retain year 1780;
   year=year+10;
   label pop='U.S. Population in Millions';
   datalines;
3929  5308  7239   9638  12866  17069  23191  31443  39818 50155
62947 75994 91972 105710 122775 131669 151325 179323 203211
226542 248710
;

proc model data=uspop;
   label a = 'Maximum Population'
         b = 'Location Parameter'
         c = 'Initial Growth Rate';
   pop = a / ( 1 + exp( b - c * (year-1790) ) );
   fit pop start=(a 1000  b 5.5  c .02)/ out=resid outresid;
run;
```

Output 20.1.1. Logistic Growth Curve Model Summary

```
                 Logistic Growth Curve Model of U.S. Population

                              The MODEL Procedure

                                 Model Summary

                          Model Variables          1
                          Parameters               3
                          Equations                1
                          Number of Statements     1


                   Model Variables  pop
                        Parameters  a(1000) b(5.5) c(0.02)
                         Equations  pop
                 Logistic Growth Curve Model of U.S. Population

                              The MODEL Procedure

                          The Equation to Estimate is

                               pop =  F(a, b, c)
```

Output 20.1.2. Logistic Growth Curve Estimation Summary

```
                 Logistic Growth Curve Model of U.S. Population

                              The MODEL Procedure

                   Nonlinear OLS Summary of Residual Errors

                 DF      DF                                           Adj
   Equation   Model   Error        SSE        MSE   R-Square         R-Sq

   pop            3      18      345.6    19.2020     0.9972       0.9969


                      Nonlinear OLS Parameter Estimates

                               Approx                  Approx
Parameter       Estimate      Std Err    t Value     Pr > |t|    Label

a               387.9307      30.0404      12.91       <.0001    Maximum Population
b               3.990385       0.0695      57.44       <.0001    Location Parameter
c               0.022703      0.00107      21.22       <.0001    Initial Growth Rate
```

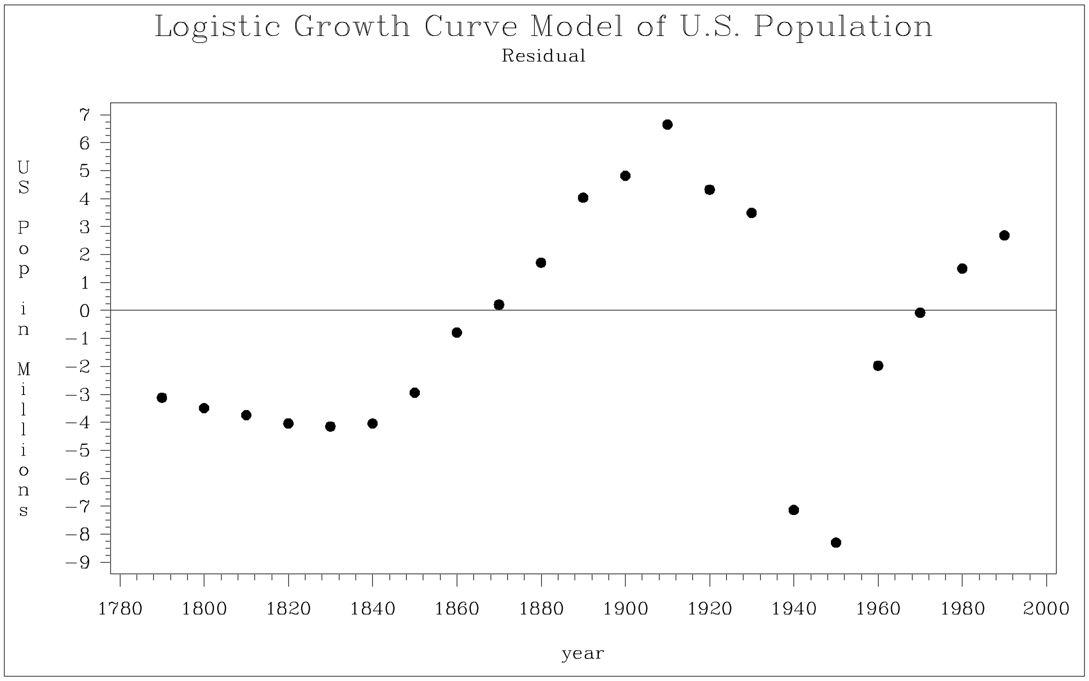
The adjusted R^2 value indicates the model fits the data well. There are only 21 observations and the model is nonlinear, so significance tests on the parameters are only approximate. The significance tests and associated approximate probabilities indicate that all the parameters are significantly different from 0.

The FIT statement included the options OUT=RESID and OUTRESID so that the residuals from the estimation are saved to the data set RESID. The residuals are plotted to check for heteroscedasticity using PROC GPLOT as follows.

```
   proc gplot data=resid;
    axis2 label=( a=-90 r=90 'US Pop in Millions' );
    plot pop*year / vref=0 vaxis=axis2
                    haxis=1780 to 2000 by 20;
    title2 "Residual";
    symbol1 v=dot;
   run;
```

The plot is shown in Output 20.1.3.

Output 20.1.3. Residual for Population Model (Actual - Predicted)



The residuals do not appear to be independent, and the model could be modified to explain the remaining nonrandom errors.

Example 20.2. A Consumer Demand Model

This example shows the estimation of a system of nonlinear consumer demand equations based on the translog functional form using seemingly unrelated regression (SUR). Expenditure shares and corresponding normalized prices are given for three goods.

Since the shares add up to one, the system is singular; therefore, one equation is omitted from the estimation process. The choice of which equation to omit is arbitrary. The nonlinear system is first estimated in unrestricted form.

```
title1 'Consumer Demand--Translog Functional Form';
title2 'Nonsymmetric Model';
proc model data=tlog1;
   endogenous share1 share2;
   parms a1 a2 b11 b12 b13 b21 b22 b23 b31 b32 b33;

   bm1 = b11 + b21 + b31;
   bm2 = b12 + b22 + b32;
   bm3 = b13 + b23 + b33;
   lp1 = log(p1);
   lp2 = log(p2);
   lp3 = log(p3);
   share1 = ( a1 + b11 * lp1 + b12 * lp2 + b13 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
   share2 = ( a2 + b21 * lp1 + b22 * lp2 + b23 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
```

```
      fit share1 share2
          start=( a1 -.14 a2 -.45 b11 .03 b12 .47 b22 .98 b31 .20
                  b32 1.11 b33 .71 ) / outsused = smatrix sur;
   run;
```

A portion of the printed output produced in the preceding example is shown in Output 20.2.1.

Output 20.2.1. Estimation Results from the Unrestricted Model

```
                    Consumer Demand--Translog Functional Form
                                Nonsymmetric Model

                               The MODEL Procedure

                                  Model Summary

                            Model Variables          5
                            Parameters              11
                            Equations                2
                            Number of Statements     8


Model Variables  share1 share2 p1 p2 p3
     Parameters  a1(-0.14) a2(-0.45) b11(0.03) b12(0.47) b13 b21
                 b22(0.98) b23 b31(0.2) b32(1.11) b33(0.71)
      Equations  share1 share2
```

```
                    Consumer Demand--Translog Functional Form
                                Nonsymmetric Model

                               The MODEL Procedure

                            The 2 Equations to Estimate

           share1 =  F(a1, b11, b12, b13, b21, b22, b23, b31, b32, b33)
           share2 =  F(a2, b11, b12, b13, b21, b22, b23, b31, b32, b33)


         NOTE: At SUR Iteration 2 CONVERGE=0.001 Criteria Met.
```

```
                  Consumer Demand--Translog Functional Form
                              Nonsymmetric Model

                             The MODEL Procedure

                  Nonlinear SUR Summary of Residual Errors

                 DF      DF                                                  Adj
Equation      Model   Error        SSE        MSE   Root MSE   R-Square     R-Sq

share1          5.5    38.5    0.00166   0.000043    0.00656     0.8067   0.7841
share2          5.5    38.5    0.00135   0.000035    0.00592     0.9445   0.9380


                      Nonlinear SUR Parameter Estimates

                                      Approx                    Approx
        Parameter       Estimate     Std Err    t Value     Pr > |t|

        a1              -0.14881     0.00225     -66.08       <.0001
        a2              -0.45776     0.00297    -154.29       <.0001
        b11             0.048382      0.0498       0.97       0.3379
        b12              0.43655      0.0502       8.70       <.0001
        b13             0.248588      0.0516       4.82       <.0001
        b21             0.586326      0.2089       2.81       0.0079
        b22             0.759776      0.2565       2.96       0.0052
        b23             1.303821      0.2328       5.60       <.0001
        b31             0.297808      0.1504       1.98       0.0550
        b32             0.961551      0.1633       5.89       <.0001
        b33               0.8291      0.1556       5.33       <.0001


             Number of Observations     Statistics for System

             Used                 44    Objective         1.7493
             Missing               0    Objective*N      76.9697
```

The model is then estimated under the restriction of symmetry (b_{ij}=b_{ji}).

Hypothesis testing requires that the **S** matrix from the unrestricted model be imposed on the restricted model, as explained in "Tests on Parameters" in this chapter. The **S** matrix saved in the data set SMATRIX is requested by the SDATA= option.

A portion of the printed output produced in the following example is shown in Output 20.2.2.

```
title2 'Symmetric Model';
proc model data=tlog1;
   var share1 share2 p1 p2 p3;
   parms a1 a2 b11 b12 b22 b31 b32 b33;
   bm1 = b11 + b12 + b31;
   bm2 = b12 + b22 + b32;
   bm3 = b31 + b32 + b33;
   lp1 = log(p1);
   lp2 = log(p2);
   lp3 = log(p3);
   share1 = ( a1 + b11 * lp1 + b12 * lp2 + b31 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
   share2 = ( a2 + b12 * lp1 + b22 * lp2 + b32 * lp3 ) /
```

```
                ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
      fit share1 share2
          start=( a1 -.14 a2 -.45 b11 .03 b12 .47 b22 .98 b31 .20
                  b32 1.11 b33 .71 ) / sdata=smatrix sur;
   run;
```

A chi-square test is used to see if the hypothesis of symmetry is accepted or rejected. (*Oc*-*Ou*) has a chi-square distribution asymptotically, where *Oc* is the constrained OBJECTIVE*N and *Ou* is the unconstrained OBJECTIVE*N. The degrees of freedom is equal to the difference in the number of free parameters in the two models.

In this example, Ou is 76.9697 and Oc is 78.4097, resulting in a difference of 1.44 with 3 degrees of freedom. You can obtain the probability value by using the following statements:

```
   data _null_;
                    /* reduced-full, nrestrictions */
      p = 1-probchi( 1.44, 3 );
      put p=;
   run;
```

The output from this DATA step run is ‘P=0.6961858724’. With this probability you cannot reject the hypothesis of symmetry. This test is asymptotically valid.

Output 20.2.2. Estimation Results from the Restricted Model

```
                     Consumer Demand--Translog Functional Form
                                  Symmetric Model

                                The MODEL Procedure

                              The 2 Equations to Estimate

                   share1 =  F(a1, b11, b12, b22, b31, b32, b33)
                   share2 =  F(a2, b11, b12, b22, b31, b32, b33)
```

```
                    Consumer Demand--Translog Functional Form
                                 Symmetric Model

                               The MODEL Procedure

                    Nonlinear SUR Summary of Residual Errors

                   DF     DF                                                  Adj
Equation        Model  Error       SSE       MSE  Root MSE  R-Square        R-Sq

share1              4     40   0.00166  0.000041   0.00644    0.8066      0.7920
share2              4     40   0.00139  0.000035   0.00590    0.9428      0.9385


                        Nonlinear SUR Parameter Estimates

                                        Approx                    Approx
        Parameter       Estimate       Std Err    t Value     Pr > |t|

        a1              -0.14684       0.00135    -108.99       <.0001
        a2               -0.4597       0.00167    -275.34       <.0001
        b11              0.02886       0.00741       3.89       0.0004
        b12             0.467827        0.0115      40.57       <.0001
        b22             0.970079        0.0177      54.87       <.0001
        b31             0.208143       0.00614      33.88       <.0001
        b32             1.102415        0.0127      86.51       <.0001
        b33             0.694245        0.0168      41.38       <.0001


             Number of Observations      Statistics for System

             Used                 44    Objective        1.7820
             Missing               0    Objective*N     78.4097
```

Example 20.3. Vector AR(1) Estimation

This example shows the estimation of a two-variable vector AR(1) error process for the Grunfeld model (Grunfeld 1960) using the %AR macro. First, the full model is estimated. Second, the model is estimated with the restriction that the errors are univariate AR(1) instead of a vector process. The following produces Output 20.3.1 and Output 20.3.2.

```
data grunfeld;
   input year gei gef gec whi whf whc;
   label gei = 'Gross Investment GE'
         gec = 'Capital Stock Lagged GE'
         gef = 'Value of Outstanding Shares GE Lagged'
         whi = 'Gross Investment WH'
         whc = 'Capital Stock Lagged WH'
         whf = 'Value of Outstanding Shares Lagged WH';
   datalines;
1935     33.1      1170.6    97.8      12.93     191.5     1.8
1936     45.0      2015.8    104.4     25.90     516.0     .8
1937     77.2      2803.3    118.0     35.05     729.0     7.4
1938     44.6      2039.7    156.2     22.89     560.4     18.1
1939     48.1      2256.2    172.6     18.84     519.9     23.5
1940     74.4      2132.2    186.6     28.57     628.5     26.5
1941     113.0     1834.1    220.9     48.51     537.1     36.2
1942     91.9      1588.0    287.8     43.34     561.2     60.8
```

```
1943     61.3      1749.4    319.9     37.02     617.2     84.4
1944     56.8      1687.2    321.3     37.81     626.7     91.2
1945     93.6      2007.7    319.6     39.27     737.2     92.4
1946     159.9     2208.3    346.0     53.46     760.5     86.0
1947     147.2     1656.7    456.4     55.56     581.4     111.1
1948     146.3     1604.4    543.4     49.56     662.3     130.6
1949     98.3      1431.8    618.3     32.04     583.8     141.8
1950     93.5      1610.5    647.4     32.24     635.2     136.7
1951     135.2     1819.4    671.3     54.38     723.8     129.7
1952     157.3     2079.7    726.1     71.78     864.1     145.5
1953     179.5     2371.6    800.3     90.08     1193.5    174.8
1954     189.6     2759.9    888.9     68.60     1188.9    213.5
;

title1 'Example of Vector AR(1) Error Process
            Using Grunfeld''s Model';
/* Note: GE stands for General Electric
            and WH for Westinghouse     */

proc model outmodel=grunmod;
   var gei whi gef gec whf whc;
   parms ge_int ge_f ge_c wh_int wh_f wh_c;
   label ge_int = 'GE Intercept'
         ge_f   = 'GE Lagged Share Value Coef'
         ge_c   = 'GE Lagged Capital Stock Coef'
         wh_int = 'WH Intercept'
         wh_f   = 'WH Lagged Share Value Coef'
         wh_c   = 'WH Lagged Capital Stock Coef';
   gei = ge_int + ge_f * gef + ge_c * gec;
   whi = wh_int + wh_f * whf + wh_c * whc;
run;
```

The preceding PROC MODEL step defines the structural model and stores it in the model file named GRUNMOD.

The following PROC MODEL step reads in the model, adds the vector autoregressive terms using %AR, and requests SUR estimation using the FIT statement.

```
title2 'With Unrestricted Vector AR(1) Error Process';
proc model data=grunfeld model=grunmod;
   %ar( ar, 1, gei whi )
   fit gei whi / sur;
run;
```

The final PROC MODEL step estimates the restricted model.

```
title2 'With restricted AR(1) Error Process';
proc model data=grunfeld model=grunmod;
   %ar( gei, 1 )
   %ar( whi, 1)
   fit gei whi / sur;
run;
```

Output 20.3.1. Results for the Unrestricted Model (Partial Output)

```
           Example of Vector AR(1) Error Process Using Grunfeld's Model
                   With Unrestricted Vector AR(1) Error Process

                               The MODEL Procedure

                                  Model Summary

                            Model Variables          6
                            Parameters              10
                            Equations                2
                            Number of Statements     6


 Model Variables  gei whi gef gec whf whc
      Parameters  ge_int ge_f ge_c wh_int wh_f wh_c ar_l1_1_1(0)
                  ar_l1_1_2(0) ar_l1_2_1(0) ar_l1_2_2(0)
       Equations  gei whi
```

```
           Example of Vector AR(1) Error Process Using Grunfeld's Model
                   With Unrestricted Vector AR(1) Error Process

                               The MODEL Procedure

                            The 2 Equations to Estimate

     gei =  F(ge_int, ge_f, ge_c, wh_int, wh_f, wh_c, ar_l1_1_1, ar_l1_1_2)
     whi =  F(ge_int, ge_f, ge_c, wh_int, wh_f, wh_c, ar_l1_2_1, ar_l1_2_2)


        NOTE: At SUR Iteration 9 CONVERGE=0.001 Criteria Met.
```

```
            Example of Vector AR(1) Error Process Using Grunfeld’s Model
                    With Unrestricted Vector AR(1) Error Process

                                The MODEL Procedure

                      Nonlinear SUR Summary of Residual Errors

                        DF      DF                                          Adj
   Equation          Model   Error        SSE        MSE   R-Square       R-Sq

   gei                   5      15     9374.5      625.0     0.7910     0.7352
   whi                   5      15     1429.2    95.2807     0.7940     0.7391


                          Nonlinear SUR Parameter Estimates

                                  Approx                 Approx
Parameter      Estimate         Std Err    t Value     Pr > |t|   Label

ge_int         -42.2858         30.5284      -1.39       0.1863   GE Intercept
ge_f           0.049894          0.0153       3.27       0.0051   GE Lagged Share
                                                                  Value Coef
ge_c           0.123946          0.0458       2.70       0.0163   GE Lagged Capital
                                                                  Stock Coef
wh_int         -4.68931          8.9678      -0.52       0.6087   WH Intercept
wh_f           0.068979          0.0182       3.80       0.0018   WH Lagged Share
                                                                  Value Coef
wh_c           0.019308          0.0754       0.26       0.8015   WH Lagged Capital
                                                                  Stock Coef
ar_l1_1_1      0.990902          0.3923       2.53       0.0233   AR(ar) gei: LAG1
                                                                  parameter for gei
ar_l1_1_2      -1.56252          1.0882      -1.44       0.1716   AR(ar) gei: LAG1
                                                                  parameter for whi
ar_l1_2_1      0.244161          0.1783       1.37       0.1910   AR(ar) whi: LAG1
                                                                  parameter for gei
ar_l1_2_2      -0.23864          0.4957      -0.48       0.6372   AR(ar) whi: LAG1
                                                                  parameter for whi
```

Output 20.3.2. Results for the Restricted Model (Partial Output)

```
            Example of Vector AR(1) Error Process Using Grunfeld’s Model
                        With Restricted AR(1) Error Process

                                The MODEL Procedure

                                   Model Summary

                             Model Variables         6
                             Parameters              8
                             Equations               2
                             Number of Statements    6


     Model Variables  gei whi gef gec whf whc
          Parameters  ge_int ge_f ge_c wh_int wh_f wh_c gei_l1(0) whi_l1(0)
           Equations  gei whi
          Example of Vector AR(1) Error Process Using Grunfeld’s Model
                        With Restricted AR(1) Error Process

                               The MODEL Procedure

                      Nonlinear SUR Summary of Residual Errors

                     DF      DF                                          Adj
   Equation       Model   Error        SSE        MSE   R-Square       R-Sq

   gei                4      16    10558.8      659.9     0.7646     0.7204
   whi                4      16     1669.8      104.4     0.7594     0.7142


                          Nonlinear SUR Parameter Estimates

                                Approx                  Approx
Parameter       Estimate       Std Err    t Value     Pr > |t|   Label

ge_int          -30.1239       29.7227      -1.01       0.3259   GE Intercept
ge_f            0.043527        0.0149       2.93       0.0099   GE Lagged Share
                                                                 Value Coef
ge_c            0.119206        0.0423       2.82       0.0124   GE Lagged Capital
                                                                 Stock Coef
wh_int          3.112671        9.2765       0.34       0.7416   WH Intercept
wh_f            0.053932        0.0154       3.50       0.0029   WH Lagged Share
                                                                 Value Coef
wh_c            0.038246        0.0805       0.48       0.6410   WH Lagged Capital
                                                                 Stock Coef
gei_l1          0.482397        0.2149       2.24       0.0393   AR(gei) gei lag1
                                                                 parameter
whi_l1          0.455711        0.2424       1.88       0.0784   AR(whi) whi lag1
                                                                 parameter
```

Example 20.4. MA(1) Estimation

This example estimates parameters for an MA(1) error process for the Grunfeld model, using both the unconditional least-squares and the maximum-likelihood methods. The ARIMA procedure estimates for Westinghouse equation are shown for comparison. The output of the following code is summarized in Output 20.4.1:

```
title1 'Example of MA(1) Error Process Using Grunfeld''s Model';
title2 'MA(1) Error Process Using Unconditional Least Squares';
proc model data=grunfeld model=grunmod;
   %ma(gei,1, m=uls);
   %ma(whi,1, m=uls);
   fit whi gei start=( gei_m1 0.8 -0.8) / startiter=2;
run;
```

Output 20.4.1. PROC MODEL Results Using ULS Estimation

```
              Example of MA(1) Error Process Using Grunfeld's Model
              MA(1) Error Process Using Unconditional Least Squares

                               The MODEL Procedure

                     Nonlinear OLS Summary of Residual Errors

                     DF      DF                                       Adj
   Equation       Model   Error        SSE        MSE   R-Square     R-Sq

   whi                4      16     1874.0      117.1     0.7299   0.6793
   resid.whi                 16     1295.6    80.9754
   gei                4      16    13835.0      864.7     0.6915   0.6337
   resid.gei                 16     7646.2      477.9


                       Nonlinear OLS Parameter Estimates

                                Approx                  Approx
Parameter      Estimate        Std Err    t Value     Pr > |t|    Label

ge_int          -26.839        32.0908      -0.84       0.4153    GE Intercept
ge_f           0.038226         0.0150       2.54       0.0217    GE Lagged Share
                                                                  Value Coef
ge_c           0.137099         0.0352       3.90       0.0013    GE Lagged Capital
                                                                  Stock Coef
wh_int         3.680835         9.5448       0.39       0.7048    WH Intercept
wh_f           0.049156         0.0172       2.85       0.0115    WH Lagged Share
                                                                  Value Coef
wh_c           0.067271         0.0708       0.95       0.3559    WH Lagged Capital
                                                                  Stock Coef
gei_m1         -0.87615         0.1614      -5.43       <.0001    MA(gei) gei lag1
                                                                  parameter
whi_m1         -0.75001         0.2368      -3.17       0.0060    MA(whi) whi lag1
                                                                  parameter
```

The estimation summary from the following PROC ARIMA statements is shown in Output 20.4.2.

```
title2 'PROC ARIMA Using Unconditional Least Squares';

proc arima data=grunfeld;
   identify var=whi cross=(whf whc ) noprint;
   estimate q=1 input=(whf whc) method=uls maxiter=40;
run;
```

Output 20.4.2. PROC ARIMA Results Using ULS Estimation

```
                    Example of MA(1) Error Process Using Grunfeld's Model
                         PROC ARIMA Using Unconditional Least Squares

                                    The ARIMA Procedure

                            Unconditional Least Squares Estimation

                                Approx Std
Parameter     Estimate         Error     t Value   Pr > |t|    Lag  Variable  Shift

MU             3.68608       9.54425        0.39     0.7044      0  whi           0
MA1,1         -0.75005       0.23704       -3.16     0.0060      1  whi           0
NUM1           0.04914       0.01723        2.85     0.0115      0  whf           0
NUM2           0.06731       0.07077        0.95     0.3557      0  whc           0


                       Constant Estimate      3.686077
                       Variance Estimate      80.97535
                       Std Error Estimate     8.998631
                       AIC                    149.0044
                       SBC                    152.9873
                       Number of Residuals          20
```

The model stored in Example 20.3 is read in using the MODEL= option and the moving average terms are added using the %MA macro.

The MA(1) model using maximum likelihood is estimated using the following:

```
   title2 'MA(1) Error Process Using Maximum Likelihood ';
   proc model data=grunfeld model=grunmod;
      %ma(gei,1, m=ml);
      %ma(whi,1, m=ml);
      fit whi gei;
   run;
```

For comparison, the model is estimated using PROC ARIMA as follows:

```
   title2 'PROC ARIMA Using Maximum Likelihood ';
   proc arima data=grunfeld;
      identify var=whi cross=(whf whc) noprint;
      estimate q=1 input=(whf whc) method=ml;
   run;
```

PROC ARIMA does not estimate systems so only one equation is evaluated.

The estimation results are shown in Output 20.4.3 and Output 20.4.4. The small differences in the parameter values between PROC MODEL and PROC ARIMA can be eliminated by tightening the convergence criteria for both procedures.

Output 20.4.3. PROC MODEL Results Using ML Estimation

```
                Example of MA(1) Error Process Using Grunfeld's Model
                    MA(1) Error Process Using Maximum Likelihood

                                 The MODEL Procedure

                      Nonlinear OLS Summary of Residual Errors

                       DF      DF                                            Adj
   Equation         Model   Error        SSE          MSE   R-Square        R-Sq

   whi                  4      16     1857.5        116.1     0.7323      0.6821
   resid.whi                   16     1344.0      84.0012
   gei                  4      16    13742.5        858.9     0.6936      0.6361
   resid.gei                   16     8095.3        506.0


                          Nonlinear OLS Parameter Estimates

                                Approx                  Approx
Parameter      Estimate    Std Err    t Value    Pr > |t|    Label

ge_int          -25.002    34.2933      -0.73      0.4765    GE Intercept
ge_f            0.03712     0.0161       2.30      0.0351    GE Lagged Share
                                                             Value Coef
ge_c           0.137788     0.0380       3.63      0.0023    GE Lagged Capital
                                                             Stock Coef
wh_int         2.946761     9.5638       0.31      0.7620    WH Intercept
wh_f           0.050395     0.0174       2.89      0.0106    WH Lagged Share
                                                             Value Coef
wh_c           0.066531     0.0729       0.91      0.3749    WH Lagged Capital
                                                             Stock Coef
gei_m1         -0.78516     0.1942      -4.04      0.0009    MA(gei) gei lag1
                                                             parameter
whi_m1         -0.69389     0.2540      -2.73      0.0148    MA(whi) whi lag1
                                                             parameter
```

Output 20.4.4. PROC ARIMA Results Using ML Estimation

```
                Example of MA(1) Error Process Using Grunfeld's Model
                         PROC ARIMA Using Maximum Likelihood

                                 The ARIMA Procedure

                           Maximum Likelihood Estimation

                          Approx Std
 Parameter    Estimate         Error     t Value  Pr > |t|    Lag  Variable  Shift

 MU            2.95645       9.20752        0.32    0.7481      0  whi           0
 MA1,1        -0.69305       0.25307       -2.74    0.0062      1  whi           0
 NUM1          0.05036       0.01686        2.99    0.0028      0  whf           0
 NUM2          0.06672       0.06939        0.96    0.3363      0  whc           0


                          Constant Estimate      2.956449
                          Variance Estimate      81.29645
                          Std Error Estimate     9.016455
                          AIC                    148.9113
                          SBC                    152.8942
                          Number of Residuals          20
```

Example 20.5. Polynomial Distributed Lags Using %PDL

This example shows the use of the %PDL macro for polynomial distributed lag models. Simulated data is generated so that Y is a linear function of six lags of X, with the lag coefficients following a quadratic polynomial. The model is estimated using a fourth-degree polynomial, both with and without endpoint constraints. The example uses simulated data generated from the following model:

$$y_t = 10 + \sum_{z=0}^{6} f(z)x_{t-z} + \epsilon$$

$$f(z) = -5z^2 + 1.5z$$

The LIST option prints the model statements added by the %PDL macro.

```
/*--------------------------------------------------------------*/
/*  Generate Simulated Data for a Linear Model with a PDL on X  */
/*        y = 10 + x(6,2) + e                                   */
/*        pdl(x) = -5.*(lg)**2 + 1.5*(lg) + 0.                  */
/*--------------------------------------------------------------*/
data pdl;
   pdl2=-5.; pdl1=1.5; pdl0=0;
   array zz(i) z0-z6;
   do i=1 to 7;
      z=i-1;
      zz=pdl2*z**2 + pdl1*z + pdl0;
      end;
   do n=-11 to 30;
      x  =10*ranuni(1234567)-5;
      pdl=z0*x + z1*xl1 + z2*xl2 + z3*xl3 + z4*xl4 + z5*xl5 + z6*xl6;
      e  =10*rannor(123);
      y  =10+pdl+e;
      if n>=1 then output;
      xl6=xl5; xl5=xl4; xl4=xl3; xl3=xl2; xl2=xl1; xl1=x;
      end;
run;

title1 'Polynomial Distributed Lag Example';

title3 'Estimation of PDL(6,4) Model-- No Endpoint Restrictions';
proc model data=pdl;
   parms int;                  /* declare the intercept parameter */
   %pdl( xpdl, 6, 4 )          /* declare the lag distribution */
   y = int + %pdl( xpdl, x );  /* define the model equation */
   fit y / list;               /* estimate the parameters */
run;
```

Output 20.5.1. PROC MODEL Listing of Generated Program

```
                       Polynomial Distributed Lag Example

           Estimation of PDL(6,4) Model-- No Endpoint Restrictions

                             The MODEL Procedure

                       Listing of Compiled Program Code
          Stmt    Line:Col      Statement as Parsed

             1    25242:14      XPDL_L0 = XPDL_0;
             2    25254:14      XPDL_L1 = XPDL_0 + XPDL_1 +
                                XPDL_2 + XPDL_3 + XPDL_4;
             3    25283:14      XPDL_L2 = XPDL_0 + XPDL_1 *
                                2 + XPDL_2 * 2 ** 2 + XPDL_3
                                * 2 ** 3 + XPDL_4 * 2 ** 4;
             4    25331:14      XPDL_L3 = XPDL_0 + XPDL_1 *
                                3 + XPDL_2 * 3 ** 2 + XPDL_3
                                * 3 ** 3 + XPDL_4 * 3 ** 4;
             5    25379:14      XPDL_L4 = XPDL_0 + XPDL_1 *
                                4 + XPDL_2 * 4 ** 2 + XPDL_3
                                * 4 ** 3 + XPDL_4 * 4 ** 4;
             6    25427:14      XPDL_L5 = XPDL_0 + XPDL_1 *
                                5 + XPDL_2 * 5 ** 2 + XPDL_3
                                * 5 ** 3 + XPDL_4 * 5 ** 4;
             7    25475:14      XPDL_L6 = XPDL_0 + XPDL_1 *
                                6 + XPDL_2 * 6 ** 2 + XPDL_3
                                * 6 ** 3 + XPDL_4 * 6 ** 4;
             8    25121:204     PRED.y = int + XPDL_L0 * x + XPDL_L1 *
                                LAG1( x ) + XPDL_L2 * LAG2( x ) +
                                XPDL_L3 * LAG3( x ) + XPDL_L4
                                * LAG4( x ) + XPDL_L5 * LAG5(
                                x ) + XPDL_L6 * LAG6( x );
             8    25121:204     RESID.y = PRED.y - ACTUAL.y;
             8    25121:204     ERROR.y = PRED.y - y;
             9    25218:15      ESTIMATE XPDL_L0, XPDL_L1, XPDL_L2,
                                XPDL_L3, XPDL_L4, XPDL_L5, XPDL_L6;
            10    25218:15      _est0 = XPDL_L0;
            11    25221:15      _est1 = XPDL_L1;
            12    25224:15      _est2 = XPDL_L2;
            13    25227:15      _est3 = XPDL_L3;
            14    25230:15      _est4 = XPDL_L4;
            15    25233:15      _est5 = XPDL_L5;
            16    25238:14      _est6 = XPDL_L6;
```

Output 20.5.2. PROC MODEL Results Specifying No Endpoint Restrictions

```
                              Polynomial Distributed Lag Example

                   Estimation of PDL(6,4) Model-- No Endpoint Restrictions

                                    The MODEL Procedure

                          Nonlinear OLS Summary of Residual Errors

                   DF      DF                                                    Adj
  Equation      Model   Error        SSE        MSE   Root MSE  R-Square        R-Sq

  y                 6      18     2070.8      115.0    10.7259    0.9998      0.9998


                              Nonlinear OLS Parameter Estimates

                                Approx                  Approx
Parameter      Estimate      Std Err    t Value     Pr > |t|    Label

int            9.621969       2.3238       4.14       0.0006
XPDL_0         0.084374       0.7587       0.11       0.9127    PDL(XPDL,6,4)
                                                                parameter for (L)**0
XPDL_1         0.749956       2.0936       0.36       0.7244    PDL(XPDL,6,4)
                                                                parameter for (L)**1
XPDL_2           -4.196       1.6215      -2.59       0.0186    PDL(XPDL,6,4)
                                                                parameter for (L)**2
XPDL_3         -0.21489       0.4253      -0.51       0.6195    PDL(XPDL,6,4)
                                                                parameter for (L)**3
XPDL_4         0.016133       0.0353       0.46       0.6528    PDL(XPDL,6,4)
                                                                parameter for (L)**4
```

The LIST output for the model without endpoint restrictions is shown in Output 20.5.1 and Output 20.5.2. The first seven statements in the generated program are the polynomial expressions for lag parameters XPDL_L0 through XPDL_L6. The estimated parameters are INT, XPDL_0, XPDL_1, XPDL_2, XPDL_3, and XPDL_4.

Portions of the output produced by the following PDL model with endpoints of the model restricted to 0 are presented in Output 20.5.3 and Output 20.5.4.

```
title3 'Estimation of PDL(6,4) Model-- Both Endpoint Restrictions';
proc model data=pdl ;
   parms int;                  /* declare the intercept parameter */
   %pdl( xpdl, 6, 4, r=both )  /* declare the lag distribution */
   y = int + %pdl( xpdl, x );  /* define the model equation */
   fit y /list;                /* estimate the parameters */
run;
```

Output 20.5.3. PROC MODEL Results Specifying Both Endpoint Restrictions

```
                        Polynomial Distributed Lag Example

            Estimation of PDL(6,4) Model-- Both Endpoint Restrictions

                               The MODEL Procedure

                    Nonlinear OLS Summary of Residual Errors

                   DF     DF                                                 Adj
  Equation      Model  Error        SSE        MSE  Root MSE  R-Square      R-Sq

  y                 4     20     449868    22493.4     150.0    0.9596    0.9535


                        Nonlinear OLS Parameter Estimates

                                 Approx                  Approx
Parameter      Estimate         Std Err    t Value     Pr > |t|   Label

int            17.08581        32.4032       0.53       0.6038
XPDL_2         13.88433         5.4361       2.55       0.0189    PDL(XPDL,6,4)
                                                                  parameter for (L)**2
XPDL_3          -9.3535         1.7602      -5.31       <.0001    PDL(XPDL,6,4)
                                                                  parameter for (L)**3
XPDL_4         1.032421         0.1471       7.02       <.0001    PDL(XPDL,6,4)
                                                                  parameter for (L)**4
```

Note that XPDL_0 and XPDL_1 are not shown in the estimate summary. They were used to satisfy the endpoint restrictions analytically by the generated %PDL macro code. Their values can be determined by back substitution.

To estimate the PDL model with one or more of the polynomial terms dropped, specify the largest degree of the polynomial desired with the %PDL macro and use the DROP= option on the FIT statement to remove the unwanted terms. The dropped parameters should be set to 0. The following PROC MODEL code demonstrates estimation with a PDL of degree 2 without the 0th order term.

```
title3 'Estimation of PDL(6,2) Model-- With XPDL_0 Dropped';
proc model data=pdl list;
   parms int;                  /* declare the intercept parameter */
   %pdl( xpdl, 6, 2 )          /* declare the lag distribution */
   y = int + %pdl( xpdl, x );  /* define the model equation */
   xpdl_0 =0;
   fit y drop=xpdl_0;          /* estimate the parameters */
run;
```

The results from this estimation are shown in Output 20.5.4.

Output 20.5.4. PROC MODEL Results Specifying %PDL(XPDL, 6, 2)

```
                      Polynomial Distributed Lag Example

               Estimation of PDL(6,2) Model-- With XPDL_0 Dropped

                              The MODEL Procedure

                   Nonlinear OLS Summary of Residual Errors

                     DF      DF                                                Adj
  Equation        Model   Error        SSE        MSE   Root MSE  R-Square      R-Sq

  y                   3      21     2114.1      100.7    10.0335    0.9998    0.9998


                      Nonlinear OLS Parameter Estimates

                                Approx                 Approx
Parameter      Estimate        Std Err    t Value    Pr > |t|    Label

int            9.536382         2.1685       4.40      0.0003
XPDL_1         1.883315         0.3159       5.96      <.0001    PDL(XPDL,6,2)
                                                                 parameter for (L)**1
XPDL_2         -5.08827         0.0656     -77.56      <.0001    PDL(XPDL,6,2)
                                                                 parameter for (L)**2
```

Example 20.6. General-Form Equations

Data for this example are generated. General-form equations are estimated and forecast using PROC MODEL. The system is a basic supply-demand model. Portions of the output from the following code is shown in Output 20.6.1 through Output 20.6.4.

```
title1 "General Form Equations for Supply-Demand Model";

proc model;
   var price quantity income unitcost;
   parms d0-d2 s0-s2;
   eq.demand=d0+d1*price+d2*income-quantity;
   eq.supply=s0+s1*price+s2*unitcost-quantity;

/* estimate the model parameters */
   fit supply demand / data=history outest=est n2sls;
   instruments income unitcost year;
run;

/* produce forecasts for income and unitcost assumptions */
   solve price quantity / data=assume out=pq;
run;

/* produce goal-seeking solutions for
     income and quantity assumptions*/
   solve price unitcost / data=goal out=pc;
run;

title2 "Parameter Estimates for the System";
proc print data=est;
run;

title2 "Price Quantity Solution";
```

```
   proc print data=pq;
   run;

   title2 "Price Unitcost Solution";
   proc print data=pc;
   run;
```

Three data sets were used in this example. The first data set, HISTORY, was used to estimate the parameters of the model. The ASSUME data set was used to produce a forecast of PRICE and QUANTITY. Notice that the ASSUME data set does not have to contain the variables PRICE and QUANTITY.

```
   data history;
      input year income unitcost price quantity;
      datalines;
   1976    2221.87    3.31220     0.17903    266.714
   1977    2254.77    3.61647     0.06757    276.049
   1978    2285.16    2.21601     0.82916    285.858
   1979    2319.37    3.28257     0.33202    295.034
   1980    2369.38    2.84494     0.63564    310.773
   1981    2395.26    2.94154     0.62011    319.185
   1982    2419.52    2.65301     0.80753    325.970
   1983    2475.09    2.41686     1.01017    342.470
   1984    2495.09    3.44096     0.52025    348.321
   1985    2536.72    2.30601     1.15053    360.750
   ;

   data assume;
      input year income unitcost;
      datalines;
   1986    2571.87    2.31220
   1987    2609.12    2.45633
   1988    2639.77    2.51647
   1989    2667.77    1.65617
   1990    2705.16    1.01601
   ;
```

The output produced by the first SOLVE statement is shown in Output 20.6.3.

The third data set, GOAL, is used in a forecast of PRICE and UNITCOST as a function of INCOME and QUANTITY.

```
   data goal;
      input year income quantity;
      datalines;
   1986    2571.87     371.4
   1987    2721.08     416.5
   1988    3327.05     597.3
   1989    3885.85     764.1
   1990    3650.98     694.3
   ;
```

The output from the final SOLVE statement is shown in Output 20.6.4.

Output 20.6.1. Printed Output from the FIT Statement

```
                    General Form Equations for Supply-Demand Model

                                 The MODEL Procedure

                              The 2 Equations to Estimate

                       supply =  F(s0(1), s1(price), s2(unitcost))
                       demand =  F(d0(1), d1(price), d2(income))
                    Instruments  1 income unitcost year
```

```
                    General Form Equations for Supply-Demand Model

                                 The MODEL Procedure

                     Nonlinear 2SLS Summary of Residual Errors

                    DF      DF                                                 Adj
Equation         Model   Error        SSE        MSE   Root MSE  R-Square     R-Sq

supply               3       7     3.3240     0.4749     0.6891
demand               3       7     1.0829     0.1547     0.3933


                          Nonlinear 2SLS Parameter Estimates

                                             Approx                  Approx
        Parameter       Estimate      Std Err    t Value     Pr > |t|

        d0              -395.887       4.1841     -94.62       <.0001
        d1              0.717328       0.5673       1.26       0.2466
        d2              0.298061      0.00187     159.65       <.0001
        s0               -107.62       4.1780     -25.76       <.0001
        s1              201.5711       1.5977     126.16       <.0001
        s2              102.2116       1.1217      91.12       <.0001
```

Output 20.6.2. Listing of OUTEST= Data Set Created in the FIT Statement

```
                    General Form Equations for Supply-Demand Model
                          Parameter Estimates for the System


                 _
                 S       _
    _  _         T       N
    N  T         A       U
    A  Y         T       S
  O M  P         U       E
  b E  E         S       D     d        d        d         s        s        s
  s _  _         _       _     0        1        2         0        1        2

  1   2SLS 0 Converged 10 -395.887 0.71733 0.29806 -107.620 201.571 102.212
```

Output 20.6.3. Listing of OUT= Data Set Created in the First SOLVE Statement

```
                  General Form Equations for Supply-Demand Model
                             Price Quantity Solution

Obs   _TYPE_     _MODE_    _ERRORS_    price    quantity    income   unitcost   year

 1    PREDICT   SIMULATE      0       1.20473    371.552   2571.87    2.31220   1986
 2    PREDICT   SIMULATE      0       1.18666    382.642   2609.12    2.45633   1987
 3    PREDICT   SIMULATE      0       1.20154    391.788   2639.77    2.51647   1988
 4    PREDICT   SIMULATE      0       1.68089    400.478   2667.77    1.65617   1989
 5    PREDICT   SIMULATE      0       2.06214    411.896   2705.16    1.01601   1990
```

Output 20.6.4. Listing of OUT= Data Set Created in the Second SOLVE Statement

```
                  General Form Equations for Supply-Demand Model
                             Price Unitcost Solution

Obs   _TYPE_     _MODE_    _ERRORS_    price    quantity    income   unitcost   year

 1    PREDICT   SIMULATE      0       0.99284     371.4    2571.87    2.72857   1986
 2    PREDICT   SIMULATE      0       1.86594     416.5    2721.08    1.44798   1987
 3    PREDICT   SIMULATE      0       2.12230     597.3    3327.05    2.71130   1988
 4    PREDICT   SIMULATE      0       2.46166     764.1    3885.85    3.67395   1989
 5    PREDICT   SIMULATE      0       2.74831     694.3    3650.98    2.42576   1990
```

Example 20.7. Spring and Damper Continuous System

This model simulates the mechanical behavior of a spring and damper system shown in Figure 20.84.

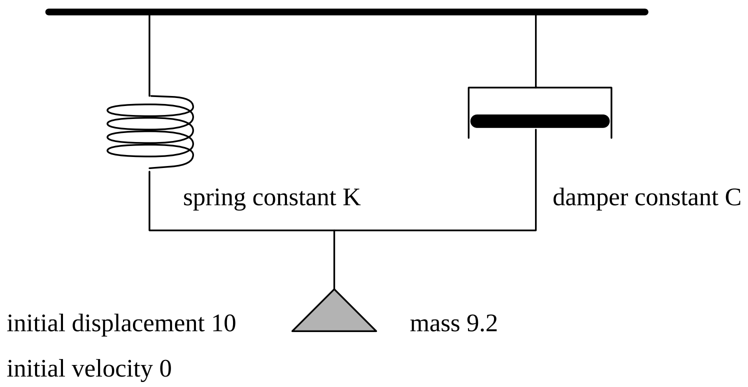


Figure 20.84. Spring and Damper System Model

A mass is hung from a spring with spring constant K. The motion is slowed by a damper with damper constant C. The damping force is proportional to the velocity, while the spring force is proportional to the displacement.

This is actually a continuous system; however, the behavior can be approximated by a discrete time model. We approximate the differential equation

$$\frac{\partial\, disp}{\partial\, time} = velocity$$

with the difference equation

$$\frac{\Delta\, disp}{\Delta\, time} = velocity$$

This is rewritten

$$\frac{disp - \mathrm{LAG(disp)}}{dt} = velocity$$

where *dt* is the time step used. In PROC MODEL, this is expressed with the program statement

```
disp = lag(disp) + vel * dt;
```

or

```
dert.disp = vel;
```

The first statement is simply a computing formula for Euler's approximation for the integral

$$disp = \int velocity\, dt$$

If the time step is small enough with respect to the changes in the system, the approximation is good. Although PROC MODEL does not have the variable step-size and error-monitoring features of simulators designed for continuous systems, the procedure is a good tool to use for less challenging continuous models.

The second form instructs the MODEL procedure to do the integration for you.

This model is unusual because there are no exogenous variables, and endogenous data are not needed. Although you still need a SAS data set to count the simulation periods, no actual data are brought in.

Since the variables DISP and VEL are lagged, initial values specified in the VAR statement determine the starting state of the system. The mass, time step, spring constant, and damper constant are declared and initialized by a CONTROL statement.

```
title1 'Simulation of Spring-Mass-Damper System';

/*- Generate some obs. to drive the simulation time periods ---*/
data one;
   do n=1 to 100;
      output;
   end;
run;
```

```
proc model data=one;
   var      force -200  disp  10  vel  0  accel -20  time 0;
   control  mass   9.2  c    1.5  dt  .1  k      20;
   force = -k * disp -c * vel;
   disp  = lag(disp) + vel * dt;
   vel   = lag(vel) + accel * dt;
   accel = force / mass;
   time  = lag(time) + dt;
```

The displacement scale is zeroed at the point where the force of gravity is offset, so the acceleration of the gravity constant is omitted from the force equation. The control variable C and K represent the damper and the spring constants respectively.

The model is simulated three times, and the simulation results are written to output data sets. The first run uses the original initial conditions specified in the VAR statement. In the second run, the initial displacement is doubled; the results show that the period of the motion is unaffected by the amplitude. In the third run, the DERT. syntax is used to do the integration. Notice that the path of the displacement is close to the old path, indicating that the original time step is short enough to yield an accurate solution. These simulations are performed by the following statements:

```
/*- Simulate the model for the base case -------------------*/
   control run '1';
   solve / out=a;
run;

/*- Simulate the model with twice the initial displacement -*/
   control run '2';
   var disp 20;
   solve / out=c;
run;

/*- Simulate the model with dert. syntax -------------*/
data two;
          do time = 0 to 10 by .2; output;end;
  run;
proc model data=two;
   var      force -200  disp  10  vel  0  accel -20  time 0;
   control  mass   9.2  c    1.5  dt  .1  k      20;
   control run '3' ;
   force = -k * disp -c * vel;
   dert.disp  = vel ;
   dert.vel   = accel;
   accel = force / mass;
   solve / out=b ;
        id time ;
run;
```

The output SAS data sets containing the solution results are merged and the displacement time paths for the three simulations are plotted. The three runs are identified on the plot as 1, 2, and 3. The following code produces Output 20.7.1 through Output 20.7.2.

```
/*- Plot the results ---------------------------------------*/
data p;
   set a b c;
run;

title2 'Overlay Plot of All Three Simulations';
proc gplot data=p;
   plot disp*time=run;
run;
```

Output 20.7.1. Printed Output Produced by PROC MODEL SOLVE Statements

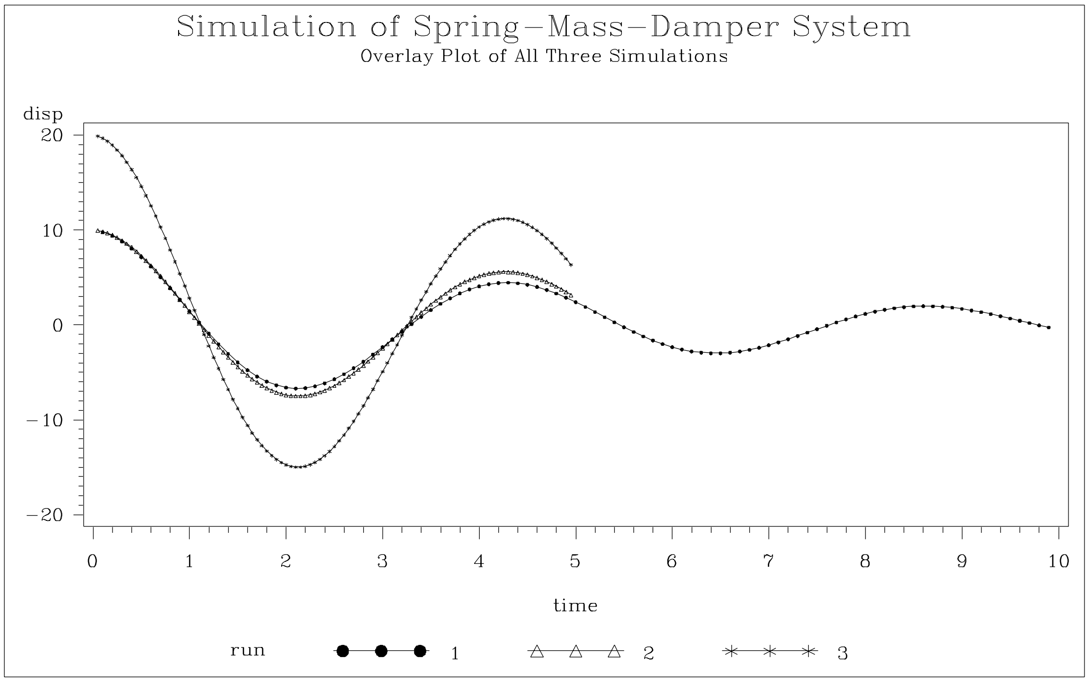
```
                     Simulation of Spring-Mass-Damper System

                               The MODEL Procedure

                                  Model Summary

                            Model Variables           5
                            Control Variables         5
                            Equations                 5
                            Number of Statements      5
                            Program Lag Length        1


          Model Variables  force(-200) disp(10) vel(0) accel(-20) time(0)
        Control Variables  mass(9.2) c(1.5) dt(0.1) k(20) run(1)
                Equations  force disp vel accel time
```

```
                  Simulation of Spring-Mass-Damper System

                          The MODEL Procedure
                     Dynamic Simultaneous Simulation

                            Data Set Options

                             DATA=    ONE
                             OUT=     A


                            Solution Summary

                  Variables Solved               5
                  Simulation Lag Length          1
                  Solution Method           NEWTON
                  CONVERGE=                   1E-8
                  Maximum CC              8.68E-15
                  Maximum Iterations             1
                  Total Iterations              99
                  Average Iterations             1


                        Observations Processed

                            Read      100
                            Lagged      1
                            Solved     99
                            First       2
                            Last      100


              Variables Solved For    force disp vel accel time
```

```
                    Simulation of Spring-Mass-Damper System

                             The MODEL Procedure
                       Dynamic Simultaneous Simulation

                               Data Set Options

                                DATA=    ONE
                                OUT=     B


                               Solution Summary

                        Variables Solved               5
                        Simulation Lag Length          1
                        Solution Method           NEWTON
                        CONVERGE=                   1E-8
                        Maximum CC              1.32E-15
                        Maximum Iterations             1
                        Total Iterations              99
                        Average Iterations             1


                            Observations Processed

                               Read      100
                               Lagged      1
                               Solved     99
                               First       2
                               Last      100


                Variables Solved For    force disp vel accel time
```

```
                    Simulation of Spring-Mass-Damper System

                             The MODEL Procedure
                      Dynamic Simultaneous Simulation

                              Data Set Options

                                DATA=    ONE
                                OUT=     C


                              Solution Summary

                     Variables Solved               5
                     Simulation Lag Length          1
                     Solution Method           NEWTON
                     CONVERGE=                   1E-8
                     Maximum CC              3.93E-15
                     Maximum Iterations             1
                     Total Iterations              99
                     Average Iterations             1


                           Observations Processed

                               Read      100
                               Lagged      1
                               Solved     99
                               First       2
                               Last      100


                Variables Solved For    force disp vel accel time
```

Output 20.7.2. Overlay Plot of all Three Simulations



Example 20.8. Nonlinear FIML Estimation

The data and model for this example were obtained from Bard (1974, p.133-138). The example is a two-equation econometric model used by Bodkin and Klein to fit U.S production data for the years 1909-1949. The model is the following:

$$g_1 = c_1 10^{c_2 z_4}(c_5 z_1^{-c_4} + (1 - c_5) z_2^{-c_4})^{-c_3/c_4} - z_3 = 0$$

$$g_2 = [c_5/(1 - c_5)](z_1/z_2)^{(-1-c_4)} - z_5 = 0$$

where z_1 is capital input, z_2 is labor input, z_3 is real output, z_4 is time in years with 1929 as year zero, and z_5 is the ratio of price of capital services to wage scale. The c_i's are the unknown parameters. z_1 and z_2 are considered

endogenous variables. A FIML estimation is performed.

```
data bodkin;
   input z1 z2 z3 z4 z5;
datalines;
1.33135 0.64629 0.4026 -20 0.24447
1.39235 0.66302 0.4084 -19 0.23454
1.41640 0.65272 0.4223 -18 0.23206
1.48773 0.67318 0.4389 -17 0.22291
1.51015 0.67720 0.4605 -16 0.22487
1.43385 0.65175 0.4445 -15 0.21879
1.48188 0.65570 0.4387 -14 0.23203
1.67115 0.71417 0.4999 -13 0.23828
1.71327 0.77524 0.5264 -12 0.26571
1.76412 0.79465 0.5793 -11 0.23410
1.76869 0.71607 0.5492 -10 0.22181
1.80776 0.70068 0.5052  -9 0.18157
1.54947 0.60764 0.4679  -8 0.22931
1.66933 0.67041 0.5283  -7 0.20595
1.93377 0.74091 0.5994  -6 0.19472
1.95460 0.71336 0.5964  -5 0.17981
2.11198 0.75159 0.6554  -4 0.18010
2.26266 0.78838 0.6851  -3 0.16933
2.33228 0.79600 0.6933  -2 0.16279
2.43980 0.80788 0.7061  -1 0.16906
2.58714 0.84547 0.7567   0 0.16239
2.54865 0.77232 0.6796   1 0.16103
2.26042 0.67880 0.6136   2 0.14456
1.91974 0.58529 0.5145   3 0.20079
1.80000 0.58065 0.5046   4 0.18307
1.86020 0.62007 0.5711   5 0.18352
1.88201 0.65575 0.6184   6 0.18847
1.97018 0.72433 0.7113   7 0.20415
2.08232 0.76838 0.7461   8 0.18847
1.94062 0.69806 0.6981   9 0.17800
1.98646 0.74679 0.7722  10 0.19979
2.07987 0.79083 0.8557  11 0.21115
```

```
2.28232 0.88462 0.9925  12 0.23453
2.52779 0.95750 1.0877  13 0.20937
2.62747 1.00285 1.1834  14 0.19843
2.61235 0.99329 1.2565  15 0.18898
2.52320 0.94857 1.2293  16 0.17203
2.44632 0.97853 1.1889  17 0.18140
2.56478 1.02591 1.2249  18 0.19431
2.64588 1.03760 1.2669  19 0.19492
2.69105 0.99669 1.2708  20 0.17912
;

proc model data=bodkin;
   parms c1-c5;
   endogenous z1 z2;
   exogenous z3 z4 z5;

   eq.g1 = c1 * 10 **(c2 * z4) * (c5*z1**(-c4)+
           (1-c5)*z2**(-c4))**(-c3/c4) - z3;
   eq.g2 = (c5/(1-c5))*(z1/z2)**(-1-c4) -z5;

   fit g1 g2 / fiml ;
run;
```

When FIML estimation is selected, the log likelihood of the system is output as the objective value. The results of the estimation are show in Output 20.8.1.

Output 20.8.1. FIML Estimation Results for U.S. Production Data

```
                              The MODEL Procedure

                  Nonlinear FIML Summary of Residual Errors

                    DF     DF                                               Adj
Equation         Model  Error        SSE        MSE  Root MSE  R-Square    R-Sq

g1                   4     37     0.0529    0.00143    0.0378
g2                   1     40     0.0173   0.000431    0.0208


                       Nonlinear FIML Parameter Estimates

                                          Approx                      Approx
        Parameter       Estimate         Std Err    t Value     Pr > |t|

        c1               0.58395          0.0218      26.76       <.0001
        c2              0.005877        0.000673       8.74       <.0001
        c3                1.3636          0.1148      11.87       <.0001
        c4              0.473688          0.2699       1.75       0.0873
        c5              0.446748          0.0596       7.49       <.0001


           Number of Observations       Statistics for System

           Used                  41    Log Likelihood     110.7773
           Missing                0
```

Example 20.9. Circuit Estimation

Consider the nonlinear circuit shown in Figure 20.85.

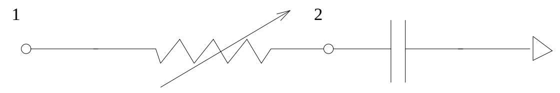


Figure 20.85. Nonlinear Resistor Capacitor Circuit

The theory of electric circuits is governed by Kirchhoff's laws: the sum of the currents flowing to a node is zero, and the net voltage drop around a closed loop is zero. In addition to Kirchhoff's laws, there are relationships between the current I through each element and the voltage drop V across the elements. For the circuit in Figure 20.85, the relationships are

$$C\frac{dV}{dt} = I$$

for the capacitor and

$$V = (R_1 + R_2(1 - \exp(-V)))I$$

for the nonlinear resistor. The following differential equation describes the current at node 2 as a function of time and voltage for this circuit:

label dvdt

$$C\frac{dV_2}{dt} - \frac{V_1 - V_2}{R_1 + R_2(1 - \exp(-V))} = 0$$

This equation can be written in the form

$$\frac{dV_2}{dt} = \frac{V_1 - V_2}{(R_1 + R_2(1 - \exp(-V)))C}$$

Consider the following data.

```
data circ;
   input v2 v1 time@@;
   datalines;
-0.00007 0.0 0.0000000001 0.00912 0.5 0.0000000002
 0.03091 1.0 0.0000000003 0.06419 1.5 0.0000000004
 0.11019 2.0 0.0000000005 0.16398 2.5 0.0000000006
 0.23048 3.0 0.0000000007 0.30529 3.5 0.0000000008
 0.39394 4.0 0.0000000009 0.49121 4.5 0.0000000010
 0.59476 5.0 0.0000000011 0.70285 5.0 0.0000000012
 0.81315 5.0 0.0000000013 0.90929 5.0 0.0000000014
 1.01412 5.0 0.0000000015 1.11386 5.0 0.0000000016
 1.21106 5.0 0.0000000017 1.30237 5.0 0.0000000018
 1.40461 5.0 0.0000000019 1.48624 5.0 0.0000000020
 1.57894 5.0 0.0000000021 1.66471 5.0 0.0000000022
;
```

You can estimate the parameters in the previous equation by using the following SAS statements:

```
proc model data=circ mintimestep=1.0e-23;
   parm R2 2000  R1 4000 C 5.0e-13;
   dert.v2 = (v1-v2)/((r1 + r2*(1-exp( -(v1-v2)))) * C);
   fit v2;
run;
```

The results of the estimation are shown in Output 20.9.1.

Output 20.9.1. Circuit Estimation

```
                              The MODEL Procedure

                        Nonlinear OLS Parameter Estimates

                                     Approx                     Approx
        Parameter       Estimate    Std Err    t Value    Pr > |t|

        R2              3002.465     1556.5       1.93      0.0688
        R1              4984.848     1504.9       3.31      0.0037
        C                  5E-13   1.01E-22    4.941E9      <.0001
```

Example 20.10. Systems of Differential Equations

The following is a simplified reaction scheme for the competitive inhibitors with recombinant human renin (Morelock et al. 1995).

$$\begin{tikzcd}
E + D \arrow[r, shift left, "k1f"] & ED \arrow[l, shift left, "k1r"] \\
+ & \\
I \arrow[d, shift left, "k2f"] & \\
EI \arrow[u, shift left, "k2r"] &
\end{tikzcd}$$

Figure 20.86. Competitive Inhibition of Recombinant Human Renin

In Figure 20.86, E= enzyme, D= probe, and I= inhibitor.

The differential equations describing this reaction scheme are

$$\frac{dD}{dt} = k1r*ED - k1f*E*D$$

$$\frac{dED}{dt} = k1f*E*D - k1r*ED$$

$$\frac{dE}{dt} = k1r*ED - k1f*E*D + k2r*EI - k2f*E*I$$

$$\frac{dEI}{dt} = k2f*E*I - k2r*EI$$

$$\frac{dI}{dt} = k2r*EI - k2f*E*I$$

For this system, the initial values for the concentrations are derived from equilibrium considerations (as a function of parameters) or are provided as known values.

The experiment used to collect the data was carried out in two ways; pre-incubation (type='disassoc') and no pre-incubation (type='assoc'). The data also contain repeated measurements. The data contain values for fluorescence F, which is a function of concentration. Since there are no direct data for the concentrations, all the differential equations are simulated dynamically.

The SAS statements used to fit this model are

```
proc model data=fit;

   parameters qf  = 2.1e8
              qb  = 4.0e9
              k2f = 1.8e5
              k2r = 2.1e-3
              l   = 0;

              k1f = 6.85e6;
              k1r = 3.43e-4;

      /* Initial values for concentrations */
   control dt 5.0e-7
           et 5.0e-8
           it 8.05e-6;

      /* Association initial values --------------*/
   if type = 'assoc' and time=0 then
      do;
         ed = 0;
            /* solve quadratic equation ----------*/
         a = 1;
         b = -(&it+&et+(k2r/k2f));
         c = &it*&et;
         ei = (-b-(((b**2)-(4*a*c))**.5))/(2*a);
         d = &dt-ed;
         i = &it-ei;
         e = &et-ed-ei;
      end;

      /* Disassociation initial values ----------*/
   if type = 'disassoc' and time=0 then
      do;
         ei = 0;
         a = 1;
         b = -(&dt+&et+(&k1r/&k1f));
         c = &dt*&et;
         ed = (-b-(((b**2)-(4*a*c))**.5))/(2*a);
         d = &dt-ed;
         i = &it-ei;
         e = &et-ed-ei;
      end;

   if time ne 0 then
      do;
         dert.d = k1r* ed  - k1f *e *d;

         dert.ed = k1f* e *d - k1r*ed;

         dert.e = k1r* ed - k1f* e * d  + k2r * ei - k2f * e *i;

         dert.ei = k2f* e *i - k2r * ei;

         dert.i = k2r * ei - k2f* e *i;

      end;

      /* L - offset between curves  */
   if type = 'disassoc' then
           F = (qf*(d-ed)) + (qb*ed) -L;
   else
           F = (qf*(d-ed)) + (qb*ed);

   Fit F / method=marquardt;
run;
```

This estimation requires the repeated simulation of a system of 42 differential equations (5 base differential equations and 36 differential equations to compute the partials with respect to the parameters).

The results of the estimation are shown in Output 20.10.1.

Output 20.10.1. Kinetics Estimation

```
                              The MODEL Procedure

                  Nonlinear OLS Summary of Residual Errors

                  DF     DF                                                  Adj
Equation       Model  Error        SSE        MSE  Root MSE  R-Square       R-Sq

f                  5    797     2525.0     3.1681    1.7799    0.9980     0.9980


                      Nonlinear OLS Parameter Estimates

                                          Approx                   Approx
        Parameter       Estimate      Std Err     t Value     Pr > |t|

        qf              2.0413E8       681443      299.55       <.0001
        qb              4.2263E9      9133179      462.74       <.0001
        k2f              6451229       867011        7.44       <.0001
        k2r             0.007808      0.00103        7.55       <.0001
        l               -5.76981       0.4138      -13.94       <.0001
```

Example 20.11. Monte Carlo Simulation

This example illustrates how the form of the error in a ODE model affects the results from a static and dynamic estimation. The differential equation studied is

$$\frac{dy}{dt} = a - ay$$

The analytical solution to this differential equation is

$$y = 1 - \exp(-\mathrm{at})$$

The first data set contains errors that are strictly additive and independent. The data for this estimation are generated by the following DATA step:

```
data drive1;
   a = 0.5;
   do iter=1 to 100;
      do time = 0 to 50;
         y = 1 - exp(-a*time) + 0.1 *rannor(123);
         output;
      end;
   end;
```

The second data set contains errors that are cumulative in form.

```
data drive2;
   a = 0.5;
   yp = 1.0 + 0.01 *rannor(123);
   do iter=1 to 100;
```

```
         do time = 0 to 50;
            y = 1 - exp(-a)*(1 - yp);
            yp = y + 0.01 *rannor(123);
            output;
         end;
      end;
```

The following statements perform the 100 static estimations for each data set:

```
   proc model data=drive1 noprint;
      parm a 0.5;
      dert.y = a - a * y;
      fit y / outest=est;
      by iter;
   run;
```

Similar code is used to produce 100 dynamic estimations with a fixed and an unknown initial value. The first value in the data set is used to simulate an error in the initial value. The following PROC UNIVARIATE code processes the estimations:

```
   proc univariate data=est noprint;
      var a;
      output out=monte mean=mean p5=p5 p95=p95;
   run;

   proc print data=monte; run;
```

The results of these estimations are summarized in Table 20.5.

Table 20.5. Monte Carlo Summary, A=0.5

Estimation	Additive Error			Cumulative Error		
Type	mean	p95	p5	mean	p95	p5
static	0.77885	1.03524	0.54733	0.57863	1.16112	0.31334
dynamic fixed	0.48785	0.63273	0.37644	3.8546E24	8.88E10	-51.9249
dynamic unknown	0.48518	0.62452	0.36754	641704.51	1940.42	-25.6054

For this example model, it is evident that the static estimation is the least sensitive to misspecification.

Example 20.12. Cauchy Distribution Estimation

In this example a nonlinear model is estimated using the Cauchy distribution. Then a simulation is done for one observation in the data.

The following DATA step creates the data for the model.

```
       /* Generate a Cauchy distributed Y */
data c;
   format date monyy.;
   call streaminit(156789);
   do t=0 to 20 by 0.1;
      date=intnx('month','01jun90'd,(t*10)-1);
      x=rand('normal');
      e=rand('cauchy') + 10 ;
      y=exp(4*x)+e;
      output;
   end;
run;
```

The model to be estimated is

$$\begin{aligned} y &= e^{-a\,x} + \epsilon \\ \epsilon &\sim \text{Cauchy}(nc) \end{aligned}$$

That is, the residuals of the model are distributed as a Cauchy distribution with non-centrality parameter nc.

The log likelihood for the Cauchy distribution is

$$like = -\log(1 + (x - nc)^2 * \pi)$$

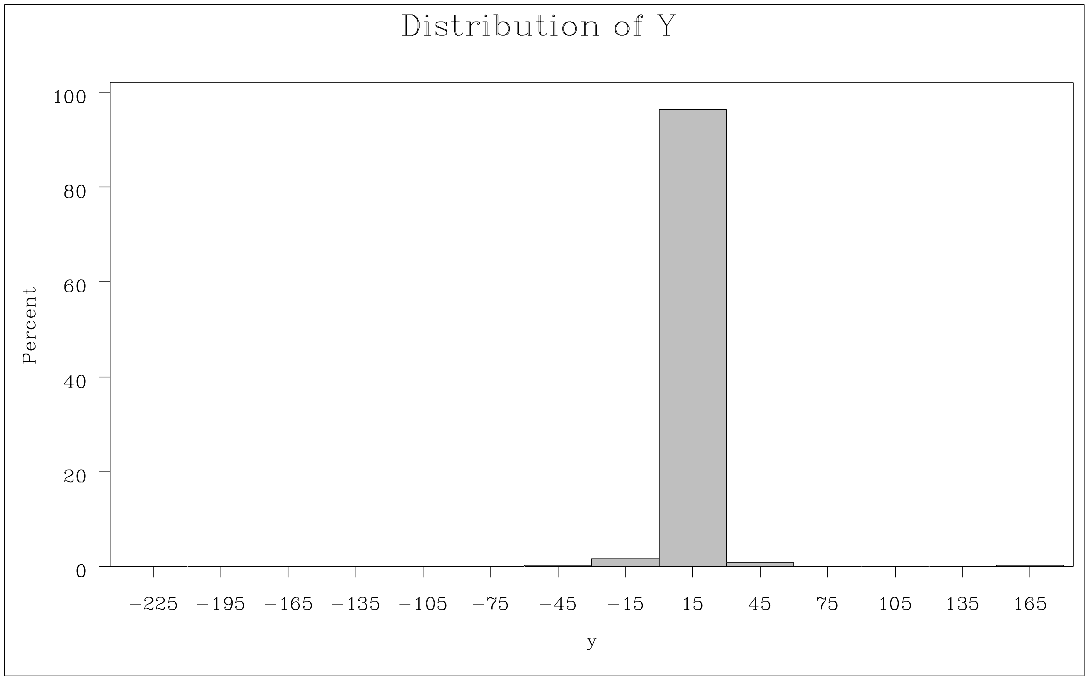
The following SAS statements specify the model and the log-likelihood function.

```
title2 'Cauchy Distribution';

proc model data=c ;
   dependent y;
   parm a -2 nc 4;
   y=exp(-a*x);

       /* Likelihood function for the residuals */
   obj = log(1+(-resid.y-nc)**2 * 3.1415926);

   errormodel y ~ general(obj) cdf=cauchy(nc);

   fit y / outsn=s1 method=marquardt;
   solve y / sdata=s1 data=c(obs=1) random=1000
             seed=256789 out=out1;
run;
```

The FIT statement uses the OUTSN= option to put out the Σ matrix for residuals from the normal distribution. The Σ matrix is 1×1 and has value 1.0 since it is a correlation matrix. The OUTS= matrix is the scalar 2989.0. Because the distribution is univariate (no covariances), the OUTS= would produce the same simulation results. The simulation is performed using the SOLVE statement.

The distribution of y is shown in the following output.

Output 20.12.1. Distribution of Y



Example 20.13. Switching Regression Example

Take the usual linear regression problem

$$y = X\beta + u$$

where *Y* denotes the *n* column vector of the dependent variable, *X* denotes the ($n \times k$) matrix of independent variables, β denotes the *k* column vector of coefficients to be estimated, *n* denotes the number of observations (*i*=1,2,...,*n*), and *k* denotes the number of independent variables.

You can take this basic equation and split it into two regimes, where the *i* th observation on *y* is generated by one regime or the other.

$$y_i = \sum_{j=1}^{k} \beta_{1j} X_{ji} + u_{1i} = x_i' \beta_1 + u_{1i}$$

$$y_i = \sum_{j=1}^{k} \beta_{2j} X_{ji} + u_{2i} = x_i' \beta_2 + u_{2i}$$

where x_{hi} and x_{hj} are the *i*th and *j*th observations, respec tively, on x_h. The errors, u_{1i} and u_{2i}, are assumed to be distributed normally and independently, with mean zero and constant variance. The variance for the first regime is σ_1^2, and the variance for the second regime is σ_2^2. If $\sigma_1^2 \neq \sigma_2^2$ and $\beta_1 \neq \beta_2$, the regression system given previously is thought to be switching between the two regimes.

The problem is to estimate β_1, β_2, σ_1, and σ_2 without knowing *a priori* which of the *n* values of the dependent variable, *y*, was generated by which regime. If it is known *a priori* which observations belong to which regime, a simple Chow test can be used to test $\sigma_1^2 = \sigma_2^2$ and $\beta_1 = \beta_2$.

Using Goldfeld and Quandt's D-method for switching regression, you can solve this problem. Assume that there exists observations on some exogenous variables z_{1i}, z_{2i}, ..., z_{pi}, where *z* determines whether the *i*th observation is generated from one equation or the other.

$$
\begin{aligned}
y_i &= x_i'\beta_1 + u_{1i} \quad \text{if} \sum_{j=1}^{p} \pi_j z_{ji} \le 0 \\
y_i &= x_i'\beta_2 + u_{2i} \quad \text{if} \sum_{j=1}^{p} \pi_j z_{ji} > 0
\end{aligned}
$$

where π_j are unknown coefficients to be estimated. Define $d(z_i)$ as a continuous approximation to a step function. Replacing the unit step function with a continuous approximation using the cumulative normal integral enables a more practical method that produces consistent estimates.

$$
d(z_i) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\sum \pi_j z_{ji}} exp\left[-\frac{1}{2} \frac{\xi^2}{\sigma^2} \right] d\xi
$$

D is the *n* dimensional diagonal matrix consisting of $d(z_i)$.

$$
D = \begin{bmatrix} d(z_1) & 0 & 0 & 0 \\ 0 & d(z_2) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & d(z_n) \end{bmatrix}
$$

The parameters to estimate are now the *k* β_1's, the *k* β_2's, σ_1^2, σ_2^2, *p* π's, and the σ introduced in the $d(z_i)$ equation. The σ can be considered as given *a priori*, or it can be estimated, in which the estimated magnitude provides an estimate of the success in discriminating between the two regimes (Goldfeld and Quandt 1976).

$$
Y = (I - D)\, X\beta_1 + DX\beta_2 + W
$$

where $W = (I - D)U_1 + DU_2$, and *W* is a vector of unob servable and heteroscedastic error terms. The covariance matrix of *W* is denoted by Ω, where $\Omega = (I - D)^2\sigma_1^2 + D^2\sigma_2^2$. The maximum likelihood parameter estimates maximize the following log-likelihood function.

$$\begin{aligned} logL &= -\frac{n}{2}\log 2\pi - \frac{1}{2}\log \mid \Omega \mid - \\ &\frac{1}{2} * \left[[Y-(I-D)X\beta_1 - DX\beta_2]'\,\Omega^{-1}\,[Y-(I-D)X\beta_1 - DX\beta_2]\right] \end{aligned}$$

As an example, you now can use this switching regression likelihood to develop a model of housing starts as a function of changes in mortgage interest rates. The data for this example is from the U.S. Census Bureau and covers the period from January 1973 to March 1999. The hypothesis is that there will be different coefficients on your model based on whether the interest rates are going up or down.

So the model for z_i will be the following

$$z_i = p * (rate_i - rate_{i-1})$$

where $rate_i$ is the mortgage interest rate at time i and p is a scale parameter to be estimated.

The regression model will be the following

$$\begin{aligned} starts_i &= intercept_1 + ar1 * starts_{i-1} + djf1 * decjanfeb \qquad z_i < 0 \\ starts_i &= intercept_2 + ar2 * starts_{i-1} + djf2 * decjanfeb \qquad z_i >= 0 \end{aligned}$$

where $starts_i$ is the number of housing starts at month i and $decjanfeb$ is a dummy variable indicating that the current month is one of December, January, or February.

This model is written using the following SAS statements.

```
proc model data=switch;
   parms sig1=10 sig2=10 int1 b11 b13 int2 b21 b23 p;
   bounds 0.0001 < sig1 sig2;

   a = p*dif(rate);       /* Upper bound of integral */
   d = probnorm(a);       /* Normal CDF as an approx of switch */

                          /* Regime 1 */
   y1 = int1 + zlag(starts)*b11 + decjanfeb *b13 ;
                          /* Regime 2 */
   y2 = int2 + zlag(starts)*b21 + decjanfeb *b23 ;
                          /* Composite regression equation */
   starts  = (1 - d)*y1 +  d*y2;

                         /* Resulting log-likelihood function */
   logL = (1/2)*( (313*log(2*3.1415)) +
        log( (sig1**2)*((1-d)**2)+(sig2**2)*(d**2) )
       + (resid.starts*( 1/( (sig1**2)*((1-d)**2)+
        (sig2**2)*(d**2) ) )*resid.starts) ) ;
   errormodel starts ~ general(logL);

   fit starts / method=marquardt converge=1.0e-5;

     /* Test for significant differences in the parms */
   test int1 = int2 ,/ lm;
   test b11 = b21 ,/ lm;
   test b13 = b23 ,/ lm;
   test sig1 = sig2 ,/ lm;

run;
```

Four TEST statements were added to test the hypothesis that the parameters were the same in both regimes. The parameter estimates and ANOVA table from this run are shown in the following output.

Output 20.13.1. Parameter Estimates from the Switching Regression

```
                              The MODEL Procedure

              Nonlinear Liklhood Summary of Residual Errors

                 DF      DF                                              Adj
Equation       Model   Error        SSE        MSE   Root MSE  R-Square     R-Sq

starts             9     304    85877.9      282.5    16.8075    0.7806   0.7748


                   Nonlinear Liklhood Parameter Estimates

                                          Approx                  Approx
        Parameter       Estimate      Std Err     t Value      Pr > |t|

        sig1            15.47451       0.9475       16.33        <.0001
        sig2            19.77797       1.2710       15.56        <.0001
        int1            32.82232       5.9070        5.56        <.0001
        b11             0.739529       0.0444       16.65        <.0001
        b13              -15.456       3.1909       -4.84        <.0001
        int2            42.73243       6.8153        6.27        <.0001
        b21             0.734112       0.0477       15.37        <.0001
        b23             -22.5178       4.2979       -5.24        <.0001
        p               25.94332       8.5181        3.05        0.0025
```

The test results shown in the following output suggest that the variance of the housing starts, SIG1 and SIG2, are significantly different in the two regimes. The tests also show a significant difference in the AR term on the housing starts.

Output 20.13.2. Parameter Estimates from the Switching Regression

```
                              The MODEL Procedure

                                 Test Results

Test            Type            Statistic    Pr > ChiSq    Label

Test0           L.M.                 0.02        0.8810    int1 = int2
Test1           L.M.               240001        <.0001    b11 = b21
Test2           L.M.                 0.02        0.8933    b13 = b23
Test3           L.M.               319354        <.0001    sig1 = sig2
```

Example 20.14. Simulating from a Mixture of Distributions

This example illustrates how to perform a multivariate simulation using models that have different error distributions. Three models are used. The first model has *t*-distributed errors. The second model is a GARCH(1,1) model with normally distributed errors. The third model has a non-central Cauchy distribution.

The following SAS statements generate the data for this example. The T and the CAUCHY data sets use a common seed so that those two series will be correlated.

```
%let df = 7.5;
%let sig1 = .5;
%let var2 = 2.5;

data t;
   format date monyy.;
   do date='1jun2001'd  to '1nov2002'd;
              /* t-distribution with df,sig1 */
      t = .05 * date + 5000  + &sig1*tinv(ranuni(1234),&df);
      output;
   end;
run;

data normal;
   format date monyy.;
   le = &var2;
   lv = &var2;
   do date='1jun2001'd  to '1nov2002'd;
              /* Normal with GARCH error structure */
      v = 0.0001 + 0.2 * le**2 + .75 * lv;
      e = sqrt( v) * rannor(12345) ;
      normal = 25 + e;
      le = e;
      lv = v;
      output;
   end;
run;

data cauchy;
   format date monyy.;
   PI = 3.1415926;
   do date='1jun2001'd  to '1nov2002'd;
      cauchy = -4 + tan((ranuni(1234) - 0.5) * PI);
      output;
   end;
run;
```

Since the multivariate joint likelihood is unknown, the models must be estimated separately. The residuals for each model are saved using the OUT= option. Also, each model is saved using the OUTMODEL= option. The ID statement is used to provide a variable in the residual data set to merge by. The XLAG function is used to model the GARCH(1,1) process. The XLAG function returns the lag of the first argument if it is nonmissing, otherwise it returns the second argument.

```
proc model data=t outmod=t;
   parms df 10 vt 4;
      t = a * date + c;
   errormodel t ~ t( vt, df );
   fit t / out=tresid;
   id date;
run;

proc model data=normal outmod=normal;
   normal = b0 ;
   h.normal = arch0 + arch1 * xlag(resid.normal **2 , mse.normal)
              + GARCH1 * xlag(h.normal, mse.normal);

   fit normal /fiml out=nresid;
   id date;
run;


proc model data= cauchy outmod=cauchy;
parms nc = 1;
     /* nc is noncentrality parm to Cauchy dist */
   cauchy = nc;
   obj = log(1+resid.cauchy**2 * 3.1415926);
   errormodel cauchy ~ general(obj) cdf=cauchy(nc);

   fit cauchy / out=cresid;
   id date;
run;
```

The simulation requires a covariance matrix created from normal residuals. The following Data Step code uses the inverse CDFs of the *t* and Cauchy distributions to convert the residuals to the normal distribution. The CORR procedure is used to create a correlation matrix using the converted residuals.

```
   /* Merge and normalize the 3 residual data sets */
data c; merge tresid nresid cresid; by date;
   t = probit(cdf("T", t/sqrt(0.2789), 16.58 ));
   cauchy = probit(cdf("CAUCHY", cauchy, -4.0623));
run;

proc corr data=c out=s;
   var t normal cauchy;
run;
```

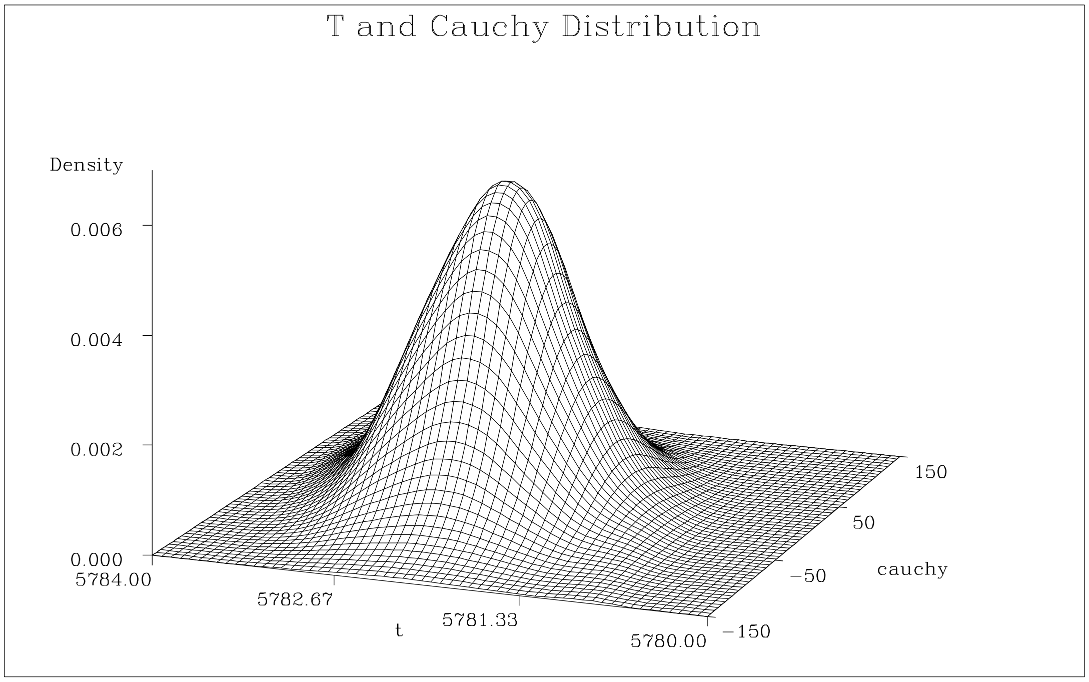
Now the models can be simulated together using the MODEL procedure SOLVE statement. The data set created by the CORR procedure is used as the correlation matrix. Note that the errormodel statement is not saved with the model and must be restated for this simulation.

```
   /* Create one observation driver data set */
data sim; merge t normal cauchy; by date;
data sim; set sim(firstobs = 519 );

proc model data=sim model=( t normal cauchy);
   errormodel t ~ t( vt, df );
   errormodel cauchy ~ cauchy(nc);
   solve t cauchy normal / random=2000 seed=1962 out=monte
      sdata=s(where=(_type_="CORR"));
run;
```

An estimation of the joint density of the *t* and Cauchy distribution is created using the KDE procedure. Bounds were placed on the Cauchy dimension because of its fat tail behavior. The joint PDF is shown in the following output.

```
proc kde gridl=5780,-150 gridu=5784,150 data=monte out=density;
   var t cauchy;
run;

title "T and Cauchy Distribution";
proc g3d data=density;
   plot t*cauchy=density;
run;
```

Output 20.14.1. Density of T and CAUCHY Truncated in the CAUCHY dimension



Example 20.15. Simple Linear Regression

This example illustrates how to use SMM to estimate a simple linear regression model for the following process:

$$y = a + bx + \epsilon, \ \epsilon \sim iid\, N(0, s^2).$$

In the following SAS code, $ysim$ is simulated and the first moment and the second moment of $ysim$ is compared with those of the observed endogenous variable y.

```
data _tmpdata;
   do i=1 to 500;
     x = rannor( 1013 );
          Y = 2 + 1.5 * x + 1.5 * rannor( 9871 );
          output;
   end;
run;

proc model data=_tmpdata;
  parms a b s;
       instrument x;

       ysim = (a+b*x) + s * rannor( 8003 );

       y = ysim;
       eq.ysq = y*y - ysim*ysim;

       fit y ysq / gmm ndraw;
       bound s > 0;

run;
```

The output of the MODEL procedure is shown in Output 20.15.1:

Output 20.15.1. PROC MODEL Output

```
                              The MODEL Procedure

                                 Model Summary

                           Model Variables          1
                           Parameters               3
                           Equations                2
                           Number of Statements     5


                             Model Variables  Y
                                  Parameters  a b s
                                   Equations  ysq Y


                          The 2 Equations to Estimate

                                 Y =  F(a(1), b(x), s)
                               ysq =  F(a, b, s)
                         Instruments  1 x



                              The MODEL Procedure

                        Nonlinear GMM Parameter Estimates

                                          Approx                  Approx
           Parameter       Estimate      Std Err    t Value     Pr > |t|

           a               2.065983       0.0657      31.45       <.0001
           b               1.511075       0.0565      26.73       <.0001
           s               1.483358       0.0498      29.78       <.0001
```

Example 20.16. AR(1) Process

This example illustrates how to use SMM to estimate an AR(1) regression model for the following process:

$$\begin{aligned} y_t &= a + bx_t + u_t, \\ u_t &= \alpha u_{t-1} + \epsilon_t, \\ \epsilon_t &\sim iid\, N(0, s^2). \end{aligned}$$

In the following SAS code, $ysim$ is simulated using this model and the endogenous variable y is set to be equal to $ysim$. The MOMENT statement creates two more moments for the estimation. One is the second moment and the other is the first order autocovariance. The NPREOBS=20 option instructs PROC MODEL to run the simulation 20 times before $ysim$ is compared to the first observation of y. Because the initial $zlag(u)$ is zero, the first $ysim$ is $a + b * x + s * rannor(8003)$. Without the NPREOBS option, this $ysim$ is matched with the first observation of y. With NPREOBS, this $ysim$, along with the next 19 $ysim$, is thrown away, and the moment match starts with the twenty-first $ysim$ with the first observation of y. This way, the initial values do not exert a large inference to the simulated endogenous variables.

```
%let nobs=500;

data _tmpdata;
   lu =0;
   do i=-10 to &nobs;
     x = rannor( 1011 );
          e = rannor( 9887 );
          u = .6 * lu + 1.5 * e;
          Y = 2 + 1.5 * x + u;
          lu = u;
          if i > 0 then output;
   end;
run;


proc model data=_tmpdata ;
  parms a b s 1 alpha .5;
       instrument x;

       u = alpha * zlag(u) + s * rannor( 8003 );

       ysim = a + b * x + u;

       y = ysim;
       eq.ysq = y*y - ysim*ysim;
       eq.ylagy = y * lag(y) - ysim * lag( ysim );

       fit y ysq ylagy / gmm npreobs=10 ndraw=10;
       bound s > 0, 1>alpha>0;

run;
```

The output of the MODEL procedure is shown in Output 20.16.1:

Output 20.16.1. PROC MODEL Output

```
                               The MODEL Procedure

                                  Model Summary

                          Model Variables         1
                          Parameters              4
                          Equations               3
                          Number of Statements    9
                          Program Lag Length      1


                      Model Variables  Y
                    Parameters(Value)  a b s(1) alpha(0.5)
                            Equations  ysq ylagy Y


                          The 3 Equations to Estimate

                             Y =  F(a(1), b(x), s, alpha)
                           ysq =  F(a, b, s, alpha)
                         ylagy =  F(a, b, s, alpha)
                     Instruments  1 x



                               The MODEL Procedure

                       Nonlinear GMM Parameter Estimates

                                            Approx                  Approx
           Parameter       Estimate      Std Err    t Value     Pr > |t|

           a               1.647842       0.1023      16.11       <.0001
           b               1.494174       0.0700      21.36       <.0001
           s               1.418301       0.0919      15.43       <.0001
           alpha           0.561595       0.0714       7.87       <.0001
```

Example 20.17. Stochastic Volatility Model

This example illustrates how to use SMM to estimate a stochastic volatility model as in Andersen and Sorensen (1996):

$$\begin{aligned} y_t &= \sigma_t z_t, \\ log(\sigma_t^2) &= a + b\,log(\sigma_{t-1}^2) + s u_t, \\ (z_t, u_t) &\sim iid\,N(0, I_2). \end{aligned}$$

This model is widely used in modeling the return process of stock prices and foreign exchange rates. This is called the stochastic volatility model because the volatility is stochastic as the random variable u_t appears in the volatility equation. The following SAS code uses three moments: absolute value, the second order moment, and absolute value of the first order autoregressive moment. Note the ADJSMMV option in the FIT statement to request the SMM covariance adjustment for the parameter estimates. Although these moments have closed form solution as shown by Andersen and Sorensen (1996), the simulation approach significantly simplifies the moment conditions.

```
%let nobs=1000;

data _tmpdata;
   a = -0.736; b=0.9; s=0.363;
   ll=sqrt( exp(a/(1-b)));;
   do i=-10 to &nobs;
     u = rannor( 101 );
          z = rannor( 98761 );
          lnssq = a+b*log(ll**2) +s*u;
          st = sqrt(exp(lnssq));
          ll = st;
          y = st * z;
          if i > 0 then output;
   end;
run;


proc model data=_tmpdata ;
  parms a b .5 s 1;
       instrument _exog_ / intonly;

       u = rannor( 8801 );
       z = rannor( 9701 );

       lsigmasq = xlag(sigmasq,exp(a));

       lnsigmasq = a + b * log(lsigmasq) + s * u;
       sigmasq = exp( lnsigmasq );

       ysim = sqrt(sigmasq) * z;

       eq.m1 = abs(y) - abs(ysim);

       eq.m2 = y**2 - ysim**2;

       eq.m5 = abs(y*lag(y))-abs(ysim*lag(ysim));

       fit m1 m2 m5 / gmm npreobs=10 ndraw=10;
       bound s > 0, 1>b>0;

run;
```

The output of the MODEL procedure is shown in Output 20.17.1.

Output 20.17.1. PROC MODEL Output

```
                              The MODEL Procedure

                                 Model Summary

                           Parameters                3
                           Equations                 3
                           Number of Statements     13
                           Program Lag Length        1


                        Parameters(Value)  a b(0.5) s(1)
                                Equations  m1 m2 m5


                            The 3 Equations to Estimate

                                    m1 =  F(a, b, s)
                                    m2 =  F(a, b, s)
                                    m5 =  F(a, b, s)
                             Instruments  1



                              The MODEL Procedure

                       Nonlinear GMM Parameter Estimates

                                               Approx                 Approx
           Parameter       Estimate           Std Err    t Value    Pr > |t|

           a               -2.28945            1.0379      -2.21      0.0276
           b               0.687496            0.1419       4.84      <.0001
           s               0.752418            0.1476       5.10      <.0001
```

Example 20.18. Duration Data Model with Unobserved Heterogeneity

All of the previous three models actually have closed form moment conditions, so the simulation approach is not necessarily required for the estimation. This example illustrates how to use SMM to estimate a model for which there is no closed form solution for the moments and thus the traditional GMM method does not apply. The model is the duration data model with unobserved heterogeneity in Gourieroux and Monfort (1993):

$$\begin{aligned} y_i &= -exp(-bx_i - \sigma u_i)log(v_i), \\ u_i &\sim N(0,1) \quad v_i \sim U_{[0,1]}. \end{aligned}$$

The SAS code is:

```
%let nobs=1000;

data _tmpdata;
   b=0.9; s=0.5;
```

```
      do i=1 to &nobs;
        u = rannor( 1011 );
              v = ranuni( 9828 );
              x = 2 * ranuni( 7621 );
              y = -exp(-b * x + s * u) * log(v);
              output;
      end;
   run;


   proc model data=_tmpdata;
     parms b .5 s 1;
          instrument x;

          u = rannor( 9871 );
          v = ranuni( 7003 );

          y = -exp(-b * x + s * u) * log(v);

          moment y = (2 3 4);

          fit y / gmm ndraw=10;

          bound s > 0, b>0;

   run;
```

The output of the MODEL procedure is shown in Output 20.18.1.

Output 20.18.1. PROC MODEL Output

```
                              The MODEL Procedure

                                 Model Summary

                           Model Variables          1
                           Parameters               2
                           Equations                4
                           Number of Statements    10


                 Model Variables  y
               Parameters(Value)  b(0.5) s(1)
                       Equations  _moment_3 _moment_2 _moment_1 y


                           The 4 Equations to Estimate

                              _moment_3 =  F(b, s)
                              _moment_2 =  F(b, s)
                              _moment_1 =  F(b, s)
                                      y =  F(b, s)
                              Instruments  1 x



                              The MODEL Procedure

                       Nonlinear GMM Parameter Estimates

                                             Approx                  Approx
           Parameter       Estimate       Std Err    t Value     Pr > |t|

           b               0.918135        0.0330      27.80       <.0001
           s               0.310181        0.0426       7.29       <.0001
```

Example 20.19. EMM Estimation of a Stochastic Volatility Model

The Efficient Method of Moments (EMM), introduced by Bansal et al. (1993 and 1995), and Gallant and Tauchen (2001), can be considered a variant of SMM. The idea is to match the efficiency of the Maximum Likelihood (ML) estimation with the flexibility of the SMM procedure. ML itself can be interpreted as a method of moments procedure, where the *score vector*, the vector of derivatives of the log-likelihood function with respect to the parameters, provides the exactly identifying moment conditions. EMM employs an auxiliary (or pseudo) model that closely matches the true model. The score vector of the auxilliary model provides the moment conditions in the SMM step.

This example uses the SMM feature of PROC MODEL to estimate the simple stochastic volatility (SV) model of Example 20.17 with the EMM method.

Suppose that your data are the time series $\{y_1, y_2, \ldots, y_n\}$, and the model that you want to estimate, or the structural model, is characterized by the vector of parameters $\boldsymbol{\theta}$. For the SV model, $\boldsymbol{\theta}$ is given by (a, b, s).

The first step of the EMM method is to fit the data with an auxiliary model (or score generator) that has transition density $f(y_t|Y_{t-1}, \boldsymbol{\eta})$, parametrized by the pseudo parameter $\boldsymbol{\eta}$, where $Y_{t-1} = \{y_{t-1}, \ldots, y_1\}$. The auxiliary model must approximate the true data generating process as closely as possible and be such that ML estimation is feasible.

The only identification requirement is that the dimension of the pseudo parameter $\boldsymbol{\eta}$ be greater than or equal to that of the structural parameter $\boldsymbol{\theta}$.

Andersen, Chung, and Sorensen (1999) showed that the GARCH(1,1) is an appropriate auxiliary model that leads to a good performance of the EMM estimator for the SV model.

The analytical expression for the GARCH(1,1) model with mean zero is

$$\begin{aligned} y_t &= \sigma_t z_t \\ \sigma_t^2 &= \omega + \alpha y_{t-1} + \beta \sigma_{t-1}^2 \end{aligned}$$

The pseudo parameter vector $\boldsymbol{\eta}$ is given by (ω, α, β).

One advantage of such a class of models is that the conditional density of y_t is Gaussian, that is,

$$f(y_t|Y_{t-1}, \boldsymbol{\eta}) \propto \frac{1}{\sigma_t} \exp\left(-\frac{y_t^2}{2\sigma_t^2}\right)$$

and therefore the score vector can easily be computed analytically.

The AUTOREG procedure provides the ML estimates, $\hat{\boldsymbol{\eta}}_n$. The output is stored in the garchout data set, while the estimates are stored in the garchest data set.

```
/*
/ estimate GARCH(1,1) model
/ ----------------------------------------------*/
proc autoreg data=_tmpdata(keep=y) outest=garchest noprint covout;
   model y =  / noint garch=(q=1,p=1) ;
   output out=garchout cev=gsigmasq r=resid;
run;
```

If the pseudo model is close enough to the structural model, in a suitable sense, Gallant and Long (1997) showed that a consistent estimator of the asymptotic covariance matrix of the sample pseudo-score vector can be obtained from the formula

$$\hat{V}_n = \frac{1}{n} \sum_{t=1}^{n} s_f(Y_t, \hat{\boldsymbol{\eta}}_n) s_f(Y_t, \hat{\boldsymbol{\eta}}_n)'$$

where $s_f(Y_t, \hat{\boldsymbol{\eta}}_n) = (\partial/\partial \boldsymbol{\eta}_n) \log f(y_t|Y_{t-1}, \hat{\boldsymbol{\eta}}_n)$ denotes the score function of the auxiliary model computed at the ML estimates.

The ML estimates of the GARCH(1,1) model are used in the following SAS statements to compute the variance-covariance matrix $\hat{V}_n$.

```
/*
/ compute the V matrix
/ ----------------------------------------------*/

data vvalues;
   set garchout(keep=y gsigmasq resid);

   /* compute scores of GARCH model */
   score_1 = (-1 + y**2/gsigmasq)/ gsigmasq;
   score_2 = (-1 + y**2/gsigmasq)*lag(gsigmasq) / gsigmasq;
   score_3 = (-1 + y**2/gsigmasq)*lag(y**2) / gsigmasq;

   array score{*} score_1-score_3;
   array v_t{*} v_t_1-v_t_6;
   array v{*} v_1-v_6;

   /* compute external product of score vector */
   do i=1 to 3;
      do j=i to 3;
         v_t{j*(j-1)/2 + i} = score{i}*score{j};
      end;
   end;

   /* average them over t */
   do s=1 to 6;
      v{s}+ v_t{s}/&nobs;
   end;
run;
```

The $\hat{V}$ matrix must be formatted to be used with the VDATA= option of the MODEL procedure. Please see the section "VDATA= Input data set" on page 1160 for more information regarding the VDATA= data set.

```
/*
/ Create a VDATA dataset acceptable to PROC MODEL
/ ------------------------------------------------ */

/* Transpose the last obs in the dataset */
proc transpose data=vvalues(firstobs=&nobs keep=v_1-v_6) out=tempv;
run;

/* Add eq and inst labels */
data vhat;
   set tempv(drop=_name_);
   value = col1;
   drop col1;
   input _type_ $ eq_row $ eq_col $ inst_row $ inst_col $;
   datalines;
      gmm m1 m1 1 1  /* intcpt is the only inst we use */
      gmm m1 m2 1 1
      gmm m2 m2 1 1
      gmm m1 m3 1 1
      gmm m2 m3 1 1
      gmm m3 m3 1 1
    ;
run;
```

The last step of the EMM procedure is to estimate $\boldsymbol{\theta}$ using SMM, where the moment conditions are given by the scores of the auxiliary model.

Given a fixed value of the parameter vector $\boldsymbol{\theta}$, and an arbitrarily large T, one can simulate a series $\{\hat{\mathrm{y}}_1(\boldsymbol{\theta}), \hat{\mathrm{y}}_2(\boldsymbol{\theta}), \ldots, \hat{\mathrm{y}}_T(\boldsymbol{\theta})\}$ from the structural model. The EMM estimator is the value $\hat{\boldsymbol{\theta}}_n$ that minimizes the quantity

$$m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n)' \hat{V}_n^{-1} m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n)$$

where

$$m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n) = \frac{1}{T} \sum_{k=1}^{T} s_f(\widehat{\mathrm{Y}}_k(\boldsymbol{\theta}), \hat{\boldsymbol{\eta}}_n)$$

is the sample moment condition evaluated at the fixed estimated pseudo parameter $\hat{\boldsymbol{\eta}}_n$. Note that the target function depends on the parameter $\boldsymbol{\theta}$ only through the simulated series $\hat{y}_k$.

The following statements generate a data set that contains $T = 20,000$ replicates of the estimated pseudo parameter $\hat{\boldsymbol{\eta}}_n$ and that is then inputed to the MODEL procedure. The EMM estimates are found using the SMM option of the FIT statement. The $\hat{V}_n$ matrix computed above serves as weighting matrix using the VDATA= option, and the scores of the GARCH(1,1) auxiliary model evaluated at the ML estimates are the moment conditions in the GMM step.

Since the number of structural parameters to estimate (3) is equal to the number of moment equations (3) times the number of instrument (1), the model is exactly identified and the objective function will have value zero at the minimum.

For simplicity, the starting values are set to the true values of the parameters.

```
/*
/ USE SMM TO FIND EMM ESTIMATES
/ ---------------------------------------------------*/

/* Generate dataset of length T */
data emm;
   set garchest(obs=1 keep = _ah_0 _ah_1 _gh_1 _mse_);
   do i=1 to 20000;
      output;
   end;
   drop i;
run;

/* Find the EMM estimates */
proc model data=emm maxiter=1000;
   parms a -0.736 b 0.9 s 0.363;
   instrument _exog_ / intonly;

      /* Describe the structural model */
      u = rannor( 8801 );
      z = rannor( 9701 );
```

```
         lsigmasq = xlag(sigmasq,exp(a));

         lnsigmasq = a + b * log(lsigmasq) + s * u;
         sigmasq = exp( lnsigmasq );

         ysim = sqrt(sigmasq) * z;

         /* Volatility of the GARCH model */
         gsigmasq = _ah_0 + _gh_1*xlag(gsigmasq, _mse_)
                    + _ah_1*xlag(ysim**2, _mse_);

         /* Use scores of the GARCH model as moment conditions */
         eq.m1 = (-1 + ysim**2/gsigmasq)/ gsigmasq;

         eq.m2 = (-1 + ysim**2/gsigmasq)*xlag(gsigmasq, _mse_) / gsigmasq;

         eq.m3 = (-1 + ysim**2/gsigmasq)*xlag(ysim**2, _mse_) / gsigmasq;

       /* Fit scores using SMM and estimated Vhat */
       fit m1 m2 m3 / gmm npreobs=10 ndraw=1 /* smm options */
                      vdata=vhat /* use estimated Vhat */
                      kernel=(bart,0,) /* turn smoothing off */;
                     bounds s > 0, 1>b>0;

   run;
```

The output of the MODEL procedure is shown in Output 20.19.1.

Output 20.19.1. PROC MODEL Output

```
                                  EMM estimates

                               The MODEL Procedure

                                  Model Summary

                            Parameters               3
                            Equations                3
                            Number of Statements    14


                   Parameters(Value)  a(-0.736) b(0.9) s(0.363)
                           Equations  m1 m2 m3


                           The 3 Equations to Estimate

                                    m1 =  F(a, b, s)
                                    m2 =  F(a, b, s)
                                    m3 =  F(a, b, s)
                             Instruments  1



                                  EMM estimates

                               The MODEL Procedure

                        Nonlinear GMM Parameter Estimates

                                           Approx                    Approx
           Parameter       Estimate       Std Err     t Value      Pr > |t|

           a                -0.5655        0.0162      -34.93        <.0001
           b               0.921023       0.00219      419.99        <.0001
           s               0.346605       0.00708       48.93        <.0001
```

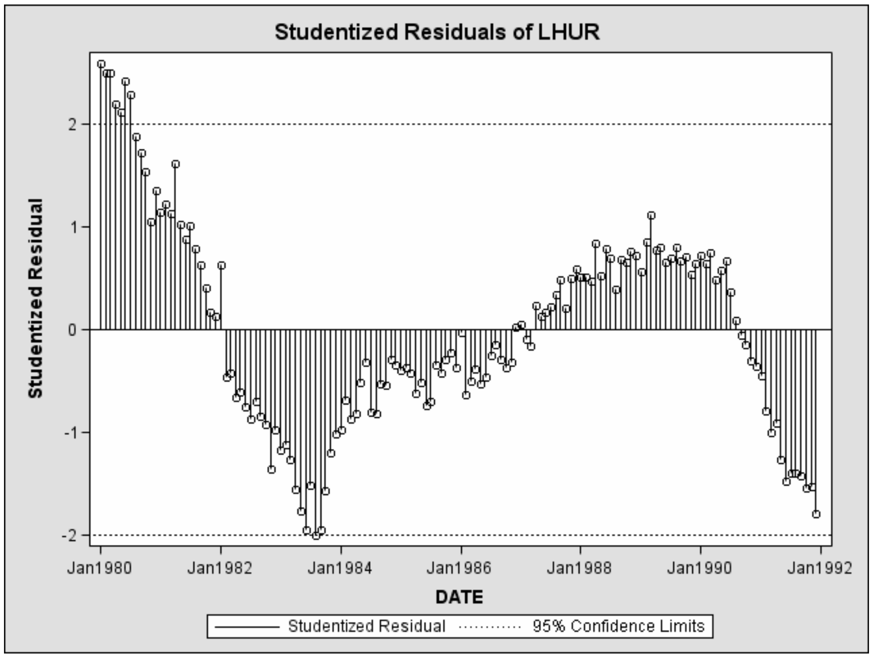
Example 20.20. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics. This is a continuation of the "Nonlinear Regression Analysis" in the section "Getting Started" on page 1003. These graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the MODEL procedure, see the "ODS Graphics" section on page 1166.

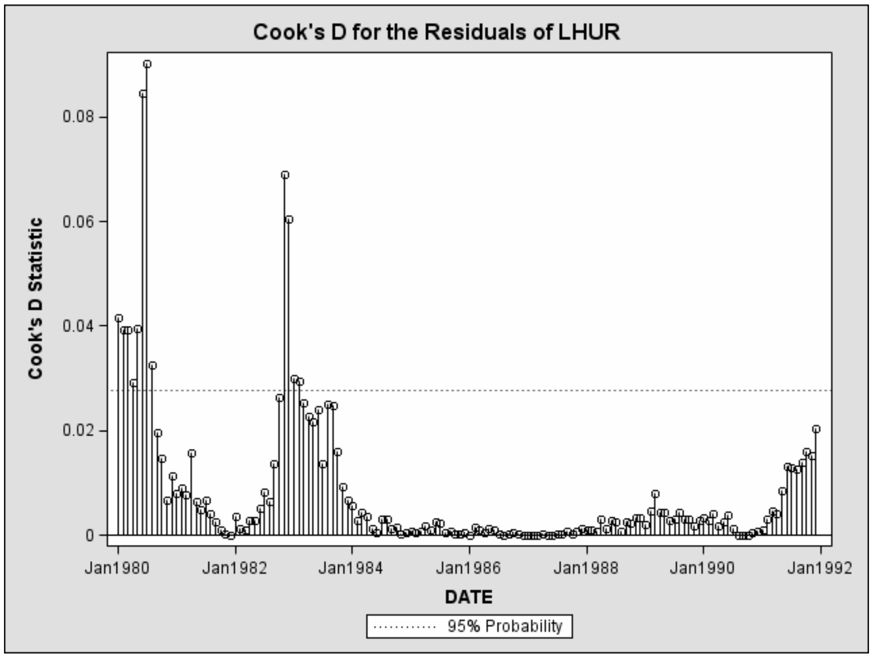
The following statements show how to generate ODS graphics plots with the MODEL procedure. The plots are displayed in Output 20.20.1 through Output 20.20.8. Note that the variable date in the ID statement is used to define the horizontal tick mark values when appropriate.

```
   ods html;
   ods graphics on;

   proc model data=sashelp.citimon;
      lhur = 1/(a * ip + b) + c;
      fit lhur;
      id date;
   run;
   quit;

   ods graphics off;
   ods html close;
```

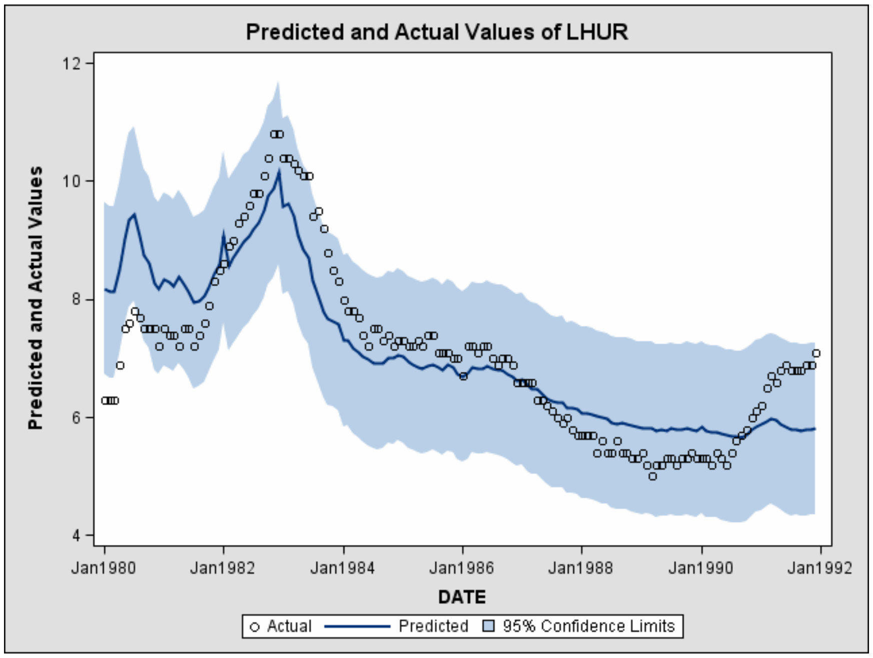
Output 20.20.1. Studentized Residuals Plot (Experimental)



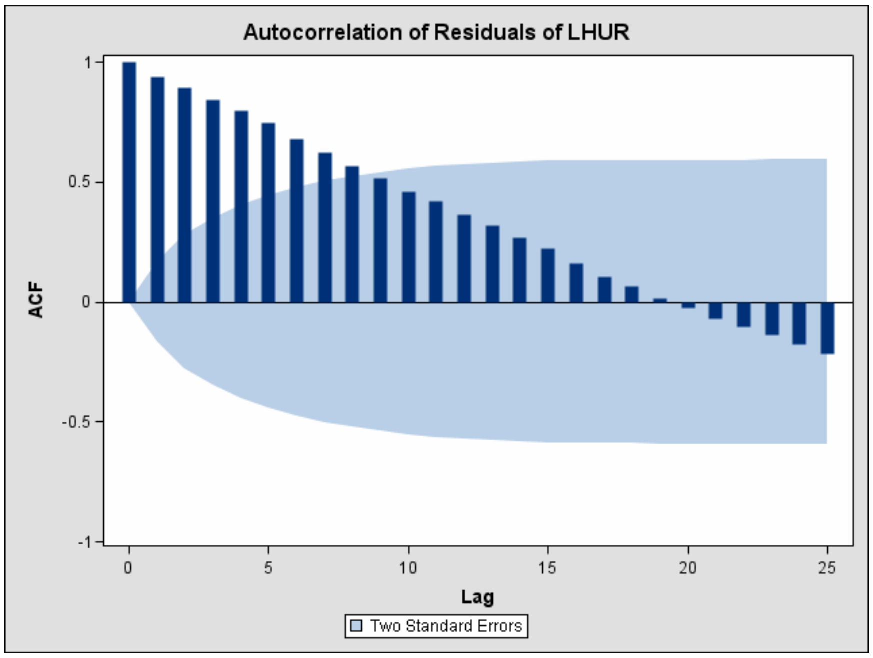
Output 20.20.2. Cook's D Plot (Experimental)



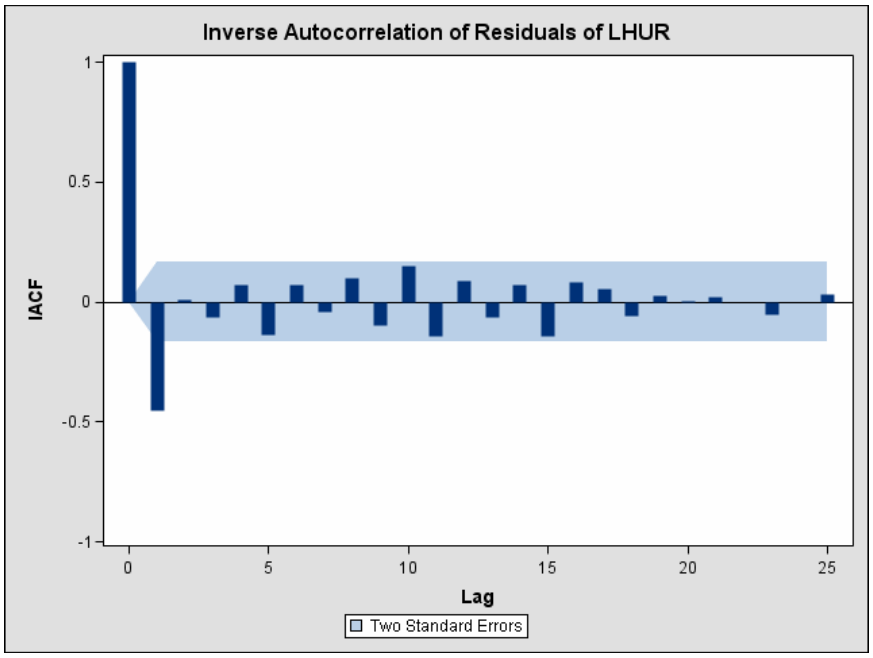
Output 20.20.3. Predicted vs Actual Plot (Experimental)



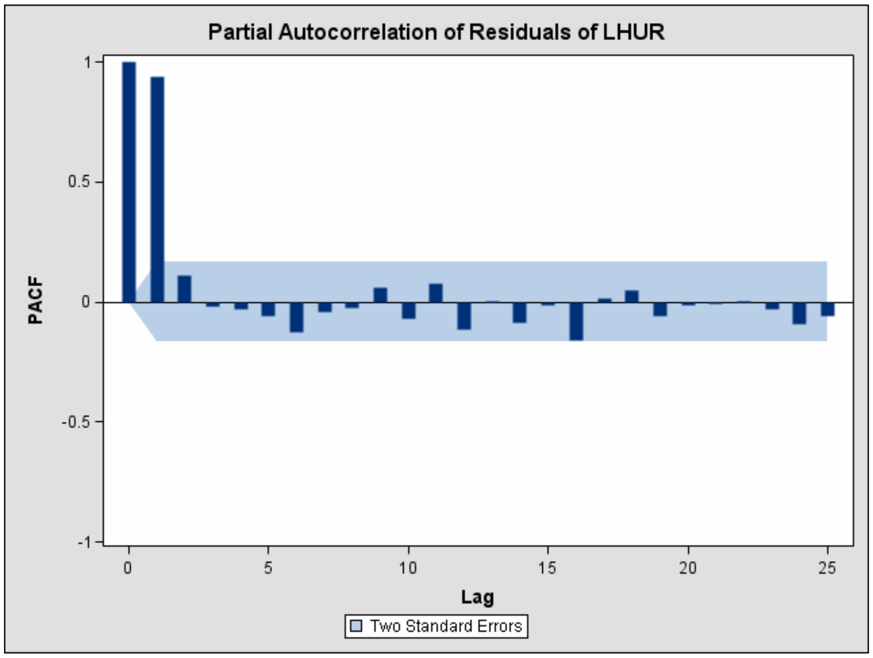
Output 20.20.4. Autocorrelation of Residuals Plot (Experimental)



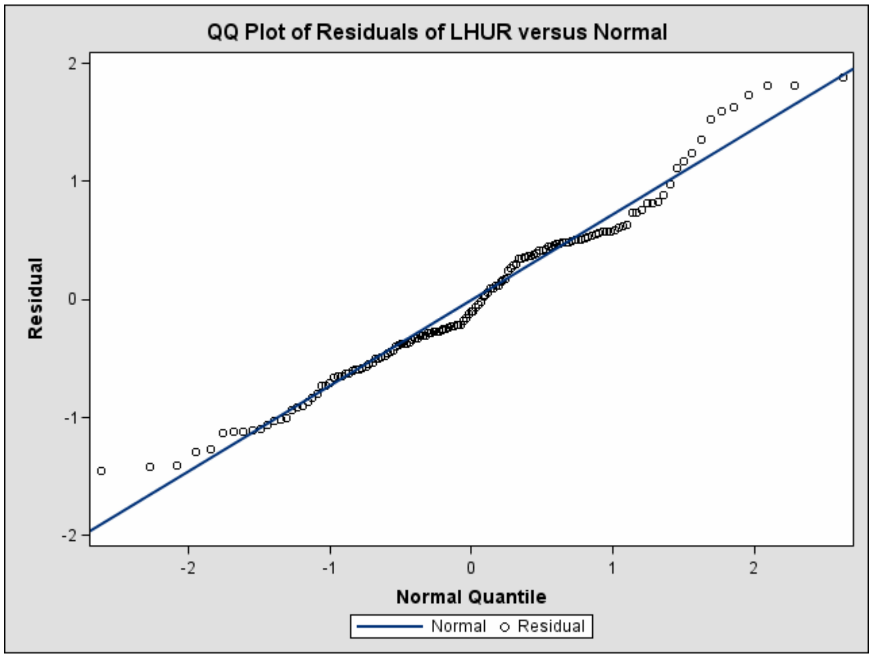
Output 20.20.5. Partial Autocorrelation of Residuals Plot (Experimental)



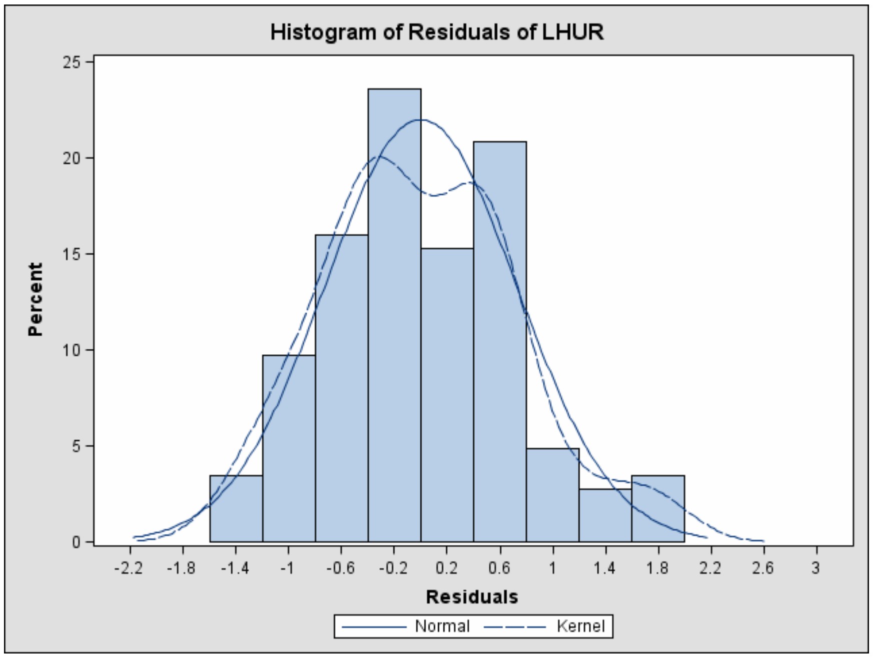
Output 20.20.6. Inverse Autocorrelation of Residuals Plot (Experimental)



Output 20.20.7. QQ Plot of Residuals (Experimental)



Output 20.20.8. Histogram of Residuals (Experimental)



References

Aiken, R.C., ed. (1985), *Stiff Computation*, New York: Oxford University Press.

Amemiya, T. (1974), “The Nonlinear Two-stage Least-squares Estimator,” *Journal of Econometrics*, 2, 105–110.

Amemiya, T. (1977), “The Maximum Likelihood Estimator and the Nonlinear Three-Stage Least Squares Estimator in the General Nonlinear Simultaneous Equation Model,” *Econometrica*, 45 (4), 955–968.

Amemiya, T. (1985), *Advanced Econometrics*, Cambridge, MA: Harvard University Press.

Andersen, T.G., Chung, H-J., and Sorensen, B.E. (1999), “Efficient Method of Moments Estimation of a Stochastic Volatility Model: A Monte Carlo Study,” *Journal of Econometrics*, 91, 61-87.

Andersen, T.G. and Sorensen, B.E. (1996), “GMM Estimation of a Stochastic Volatility Model: A Monte Carlo Study,” *Journal of Business and Economic Statistics*, 14, 328–352.

Andrews, D.W.K. (1991), “Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation,” *Econometrica*, 59 (3), 817–858.

Andrews, D.W.K. and Monahan, J.C. (1992), “Improved Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimator,” *Econometrica*, 60 (4), 953–966.

Bard, Yonathan (1974), *Nonlinear Parameter Estimation*, New York: Academic Press, Inc.

Bansal, R., Gallant, A.R., Hussey, R., and Tauchen, G.E. (1993), “Computational Aspects of Nonparametric Simulation Estimation,” Belsey, D.A., ed., *Computational Techniques for Econometrics and Economic Analysis*. Boston, MA: Kluwer Academic Publishers, 3-22.

Bansal, R., Gallant, A.R., Hussey, R., and Tauchen, G.E. (1995), “Nonparametric Estimation of Structural Models for High-Frequency Currency Market Data,” *Journal of Econometrics*, 66, 251-287.

Bates, D.M. and Watts, D.G. (1981), “A Relative Offset Orthogonality Convergence Criterion for Nonlinear Least Squares,” *Technometrics*, 23 (2), 179–183.

Belsley, D.A., Kuh, E., and Welsch, R.E. (1980), *Regression Diagnostics*, New York: John Wiley & Sons, Inc.

Binkley, J.K. and Nelson, G. (1984), “Impact of Alternative Degrees of Freedom Corrections in Two and Three Stage Least Squares,” *Journal of Econometrics*, 24 (3) 223–233.

Bowden, R.J. and Turkington, D.A. (1984), *Instrumental Variables*, Cambridge: Cambridge University Press.

Bratley, P., Fox, B.L., and H. Niederreiter (1992), “Implementation and Tests of Low-Discrepancy Sequences,” *ACM Transactions on Modeling and Computer Simulation*, 2 (3), 195-213.

Breusch, T.S. and Pagan, A.R., (1979), “A Simple Test for Heteroscedasticity and Random Coefficient Variation,” *Econometrica*, 47 (5), 1287–1294.

Breusch, T.S. and Pagan, A.R. (1980), “The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics,” *Review of Econometric Studies*, 47, 239–253.

Byrne, G.D. and Hindmarsh, A.C. (1975), “A Polyalgorithm for the Numerical Solution of ODEs,” *ACM TOMS*, 1 (1), 71–96.

Calzolari, G. and Panattoni, L. (1988), “Alternative Estimators of FIML Covariance Matrix: A Monte Carlo Study,” *Econometrica*, 56 (3), 701–714.

Chan, K.C., Karolyi, G.A., Longstaff, F.A., and Sanders, A.B. (1992), “An Empirical Comparison of Alternate Models of the Short-Term Interest Rate”, *The Journal of Finance*, 47 (3), 1209–1227.

Christensen, L.R., Jorgenson, D.W. and L.J. Lau (1975), “Transcendental Logarithmic Utility Functions,” *American Economic Review*, 65, 367–383.

Dagenais, M.G. (1978), “The Computation of FIML Estimates as Iterative Generalized Least Squares Estimates in Linear and Nonlinear Simultaneous Equation Models,” *Econometrica*, 46, 6, 1351–1362.

Davidian, M and Giltinan, D.M. (1995), *Nonlinear Models for Repeated Measurement Data*, London: Chapman & Hall.

Davidson, R. and MacKinnon, J.G. (1993), *Estimation and Inference in Econometrics*, New York: Oxford University Press.

Duffie, D. and Singleton, K.J. (1993), “Simulated Moments Estimation of Markov Models of Asset Prices,” *Econometrica* 61, 929-952.

Fair, R.C. (1984), *Specification, Estimation, and Analysis of Macroeconometric Models*, Cambridge: Harvard University Press.

Ferson, Wayne E. and Foerster, Stephen R. (1993), “Finite Sample Properties of the Generalized Method of Moments in Tests of Conditional Asset Pricing Models,” Working Paper No. 77, University of Washington.

Fox, B.L. (1986), “Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators,” *ACM Transactions on Mathematical Software,* 12 (4), 362-276.

Gallant, A.R. (1977), “Three-Stage Least Squares Estimation for a System of Simultaneous, Nonlinear, Implicit Equations,” *Journal of Econometrics*, 5, 71–88.

Gallant, A.R. (1987), *Nonlinear Statistical Models*, New York: John Wiley and Sons, Inc.

Gallant, A.R. and Holly, A. (1980), “Statistical Inference in an Implicit, Nonlinear, Simultaneous Equation Model in the Context of Maximum Likelihood Estimation,” *Econometrica*, 48 (3), 697–720.

Gallant, A.R. and Jorgenson, D.W. (1979), “Statistical Inference for a System of Simultaneous, Nonlinear, Implicit Equations in the Context of Instrumental Variables Estimation,” *Journal of Econometrics*, 11, 275–302.

Gallant, A.R. and Long, J. (1997). “Estimating Stochastic Differential Equations Efficiently by Minimum Chi-squared,” *Biometrika*, 84, 125-141.

Gallant, A.R. and Tauchen, G.E. (2001), “Efficient Method of Moments,” Working Paper, [http://www.econ.duke.edu/∼get/wpapers/ee.pdf] accessed 12 September 2001.

Gill, P.E., Murray, W., and Wright, M.H. (1981), *Practical Optimization*, New York: Academic Press Inc.

Godfrey, L.G. (1978a), “Testing Against General Autoregressive and Moving Average Error Models When the Regressors Include Lagged Dependent Variables,” *Econometrica*, 46, 1293–1301.

Godfrey, L.G. (1978b), “Testing for Higher Order Serial Correlation in Regression Equations When the Regressors Include Lagged Dependent Variables,” *Econometrica*, 46, 1303–1310.

Goldfeld, S.M. and Quandt, R.E. (1972), *Nonlinear Methods in Econometrics*, Amsterdam: North-Holland Publishing Company.

Goldfeld, S.M. and Quandt, R.E. (1973), “A Markov Model for Switching Regressions,” *Journal of Econometrics* , 3-16.

Goldfeld, S.M. and Quandt, R.E. (1973), "The Estimation of Structural Shifts by Switching Regressions," *Annals of Economic and Social Measurement*, 2/4.

Goldfeld, S.M. and Quandt, R.E. (1976), *Studies in Nonlinear Estimation*, Cambridge, MA: Ballinger Publishing Company.

Goodnight, J.H. (1979), "A Tutorial on the SWEEP Operator," *The American Statistician*, 33, 149–158.

Gourieroux, C. and Monfort, A. (1993), "Simulation Based Inference: A Survey with Special Reference to Panel Data Models," *Journal of Econometrics*, 59, 5-33.

Greene, William H. (1993), *Econometric Analysis,* New York: Macmillian Publishing Company Inc.

Gregory, A.W. and Veall, M.R. (1985), "On Formulating Wald Tests for Nonlinear Restrictions," *Econometrica*, 53, 1465–1468.

Grunfeld, Y. and Griliches, "Is Aggregation Necessarily Bad ?" *Review of Economics and Statistics,* February 1960, 113–134.

Hansen, L.P. (1982), "Large Sample Properties of Generalized Method of Moments Estimators," *Econometrica*, 50 (4), 1029–1054.

Hansen, L.P. (1985), "A Method for Calculating Bounds on the Asymptotic Covariance Matrices of Generalized Method Of Moments Estimators," *Journal of Econometrics*, 30, 203–238.

Hatanaka, M. (1978), "On the Efficient Estimation Methods for the Macro-Economic Models Nonlinear in Variables," *Journal of Econometrics*, 8, 323–356.

Hausman, J. A. (1978), "Specification Tests in Econometrics," *Econometrica,* 46(6), 1251–1271.

Hausman, J.A. and Taylor, W.E. (1982), "A Generalized Specification Test," *Economics Letters*, 8, 239–245.

Henze, N. and Zirkler, B. (1990), "A Class of Invariant Consistent tests for Multivariate Normality," *Commun. Statist. - Theory Meth.*, 19 (10), 3595–3617.

Johnston, J. (1984), *Econometric Methods,* Third Edition, New York: McGraw-Hill Book Co.

Jorgenson, D.W. and Laffont, J. (1974), "Efficient Estimation of Nonlinear Simultaneous Equations with Additive Disturbances," *Annals of Social and Economic Measurement*, 3, 615–640.

Joy, C., Boyle, P.P., and Tan, K.S. (1996), "Quasi-Monte Carlo Methods in Numerical Finance ", *Management Science*, 42 (6), 926-938.

LaMotte, L.R. (1994), "A Note on the Role of Independence in t Statistics Constructed From Linear Statistics in Regression Models," *The American Statistician*, 48 (3), 238–239.

Lee, B. and Ingram, B. (1991), "Simulation Estimation of Time Series Models," *Journal of Econometrics*, 47, 197-205.

MacKinnon, J.G, and White H. (1985), “Some Heteroskedasticity Consistent Covariance Matrix Estimators with Improved Finite Sample Properties,” *Journal of Econometrics*, 29, 305-325.

Maddala, G.S. (1977), *Econometrics*, New York: McGraw-Hill Book Co.

Mardia, K. V. (1980), “Measures of Multivariate Skewness and Kurtosis with Applications,” *Biometrika* 57 (3), 519–529.

Mardia, K. V. (1974), “Applications of Some Measures of Multivariate Skewness and Kurtosis in Testing Normality and Robustness Studies,” *The Indian Journal of Statistics* 36 (B) pt. 2, 115–128.

Matis, J.H., Miller, T.H., and Allen, D.M. (1991), *Metal Ecotoxicology Concepts and Applications*, ed. M.C Newman and A. W. McIntosh, Chelsea, MI; Lewis Publishers Inc.

Matsumoto, M. and Nishimura, T. (1998), “Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator,” *ACM Transactions on Modeling and Computer Simulation*, 8, 3-30.

McFadden, D. (1989), “A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration,” *Econometrica* 57, 995-1026.

Messer, K., and White, H. (1994), “A Note on Computing the Heteroskedasticity Consistent Covariance Matrix Using Instrumental Variable Techniques,” *Oxford Bulletin of Economics and Statistics*, 46, 181-184.

Mikhail, W.M. (1975), “A Comparative Monte Carlo Study of the Properties of Economic Estimators,” *Journal of the American Statistical Association*, 70, 94–104.

Miller, D.M. (1984), “Reducing Transformation Bias in Curve Fitting,” *The American Statistician*, 38 (2), 124–126.

Morelock, M.M., Pargellis, C.A., Graham, E.T., Lamarre, D., and Jung, G. (1995), “Time-Resolved Ligand Exchange Reactions: Kinetic Models for competitive Inhibitors with Recombinant Human Renin,” *Journal of Medical Chemistry*, 38, 1751–1761.

Nelsen, Roger B. (1999), *Introduction to Copulas*, New York, New York: Springer-Verlag.

Newey, W.K, and West, D. W. (1987), “A Simple, Positive Semi-Definite, Heteroscedasticity and Autocorrelation Consistent Covariance Matrix,” *Econometrica*, 55, 703–708.

Noble, B. and Daniel, J.W. (1977), *Applied Linear Algebra*, Englewood Cliffs, NJ: Prentice-Hall.

Ortega, J. M. and Rheinbolt, W.C. (1970), “Iterative Solution of Nonlinear Equations in Several Variables,” Burlington, MA: Academic Press.

Pakes, A. and Pollard, D. (1989), “Simulation and the Asymptotics of Optimization Estimators,” *Econometrica* 57, 1027-1057.

Parzen, E. (1957), “On Consistent Estimates of the Spectrum of a Stationary Time Series,” *Annals of Mathematical Statistics*, 28, 329–348.

Pearlman, J. G. (1980), “An Algorithm for Exact Likelihood of a High-Order Autoregressive-Moving Average Process,” *Biometrika*, 67 (1), 232–233.

Petzold, L.R. (1982), “Differential/Algebraic Equations Are Not ODEs,” *Siam J. Sci. Stat. Comput.*, 3, 367–384.

Phillips, C.B. and Park, J.Y. (1988), “On Formulating Wald Tests of Nonlinear Restrictions,” *Econometrica*, 56, 1065–1083.

Pindyck, R.S. and Rubinfeld, D.L. (1981), *Econometric Models and Economic Forecasts*, Second Edition, New York: McGraw-Hill Book Co.

Savin, N.E. and White, K.J. (1978), “Testing for Autocorrelation with Missing Observations,” *Econometrics*, 46, 59–67.

Sobol, I.M., *A Primer for the Monte Carlo Method*, Boca Raton, FL: CRC Press, 1994.

Srivastava, V. and Giles, D.E.A., (1987), “Seemingly Unrelated Regression Equation Models,” New York: Marcel Dekker, Inc.

Theil, H. (1971), *Principles of Econometrics*, New York: John Wiley & Sons, Inc.

Thursby, J., (1982), “Misspecification, Heteroscedasticity, and the Chow and Goldfield-Quandt Test,” *Review of Economics and Statistics*, 64, 314–321.

Venzon, D.J. and Moolgavkar, S.H. (1988), “A Method for Computing Profile-Likelihood Based Confidence Intervals,” *Applied Statistics*, 37, 87–94.

White, Halbert, (1980), “A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity,” *Econometrica*, 48 (4), 817–838.

Wu, D. M. (July 1973), “Alternative Tests of Independence Between Stochastic Regressors and Disturbances,” *Econometrica* , 41 (4), 733–750.

Chapter 21
The PDLREG Procedure

Chapter Contents

Chapter 21
The PDLREG Procedure

Overview

The PDLREG procedure estimates regression models for time series data in which the effects of some of the regressor variables are distributed across time. The distributed lag model assumes that the effect of an input variable X on an output Y is distributed over time. If you change the value of X at time *t*, Y will experience some immediate effect at time *t*, and it will also experience a delayed effect at times $t+1$, $t+2$, and so on up to time $t+p$ for some limit *p*.

The regression model supported by PROC PDLREG can include any number of regressors with distribution lags and any number of covariates. (Simple regressors without lag distributions are called covariates.) For example, the two-regressor model with a distributed lag effect for one regressor is written

$$y_t = \alpha + \sum_{i=0}^{p} \beta_i x_{t-i} + \gamma z_t + u_t$$

Here, x_t is the regressor with a distributed lag effect, z_t is a simple covariate, and u_t is an error term.

The distribution of the lagged effects is modeled by Almon lag polynomials. The coefficients b_i of the lagged values of the regressor are assumed to lie on a polynomial curve. That is,

$$b_i = \alpha_0^* + \sum_{j=1}^{d} \alpha_j^* i^j$$

where $d (\leq p)$ is the degree of the polynomial. For the numerically efficient estimation, the PDLREG procedure uses *orthogonal polynomials*. The preceding equation can be transformed into orthogonal polynomials.

$$b_i = \alpha_0 + \sum_{j=1}^{d} \alpha_j f_j(i)$$

where $f_j(i)$ is a polynomial of degree *j* in the lag length *i*, and α_j is a coefficient estimated from the data.

The PDLREG procedure supports endpoint restrictions for the polynomial. That is, you can constrain the estimated polynomial lag distribution curve so that $b_{-1} = 0$ or

$b_{p+1} = 0$, or both. You can also impose linear restrictions on the parameter estimates for the covariates.

You can specify a minimum degree and a maximum degree for the lag distribution polynomial, and the procedure fits polynomials for all degrees in the specified range. (However, if distributed lags are specified for more that one regressor, you can specify a range of degrees for only one of them.)

The PDLREG procedure can also test for autocorrelated residuals and perform autocorrelated error correction using the autoregressive error model. You can specify any order autoregressive error model and can specify several different estimation methods for the autoregressive model, including exact maximum likelihood.

The PDLREG procedure computes generalized Durbin-Watson statistics to test for autocorrelated residuals. For models with lagged dependent variables, the procedure can produce Durbin *h* and Durbin *t* statistics. You can request significance level *p*-values for the Durbin-Watson, Durbin *h*, and Durbin *t* statistics. See Chapter 12, “The AUTOREG Procedure,” for details about these statistics.

The PDLREG procedure assumes that the input observations form a time series. Thus, the PDLREG procedure should be used only for ordered and equally spaced time series data.

Getting Started

Use the MODEL statement to specify the regression model. The PDLREG procedure’s MODEL statement is written like MODEL statements in other SAS regression procedures, except that a regressor can be followed by a lag distribution specification enclosed in parentheses.

For example, the following MODEL statement regresses Y on X and Z and specifies a distributed lag for X:

```
model y = x(4,2) z;
```

The notation X(4,2) specifies that the model includes X and 4 lags of X, with the coefficients of X and its lags constrained to follow a second-degree (quadratic) polynomial. Thus, the regression model specified by this MODEL statement is

$$y_t = a + b_0 x_t + b_1 x_{t-1} + b_2 x_{t-2} + b_3 x_{t-3} + b_4 x_{t-4} + c z_t + u_t$$

$$b_i = \alpha_0 + \alpha_1 f_1(i) + \alpha_2 f_2(i)$$

where $f_1(i)$ is a polynomial of degree 1 in *i* and $f_2(i)$ is a polynomial of degree 2 in *i*.

Lag distribution specifications are enclosed in parentheses and follow the name of the regressor variable. The general form of the lag distribution specification is

regressor-name **(** *length, degree, minimum-degree, end-constraint* **)**

where:

length	is the length of the lag distribution; that is, the number of lags of the regressor to use
degree	is the degree of the distribution polynomial
minimum-degree	is an optional minimum degree for the distribution polynomial
end-constraint	is an optional endpoint restriction specification, which can have the values FIRST, LAST, or BOTH

If the *minimum-degree* option is specified, the PDLREG procedure estimates models for all degrees between *minimum-degree* and *degree*.

Introductory Example

The following statements generate simulated data for variables Y and X. Y depends on the first three lags of X, with coefficients .25, .5, and .25. Thus, the effect of changes of X on Y takes effect 25% after one period, 75% after two periods, and 100% after three periods.

```
data test;
   xl1 = 0; xl2 = 0; xl3 = 0;
   do t = -3 to 100;
      x = ranuni(1234);
      y = 10 + .25 * xl1 + .5 * xl2 + .25 * xl3 + .1 * rannor(1234);
      if t > 0 then output;
      xl3 = xl2; xl2 = xl1; xl1 = x;
      end;
run;
```

The following statements use the PDLREG procedure to regress Y on a distributed lag of X. The length of the lag distribution is 4, and the degree of the distribution polynomial is specified as 3.

```
proc pdlreg data=test;
   model y = x( 4, 3 );
run;
```

The PDLREG procedure first prints a table of statistics for the residuals of the model, as shown in Figure 21.1. See Chapter 12 for an explanation of these statistics.

```
                              The PDLREG Procedure

                            Dependent Variable    y


                        Ordinary Least Squares Estimates

         SSE                   0.86604442    DFE                      91
         MSE                      0.00952    Root MSE            0.09755
         SBC                   -156.72612    AIC              -169.54786
         Regress R-Square          0.7711    Total R-Square       0.7711
         Durbin-Watson             1.9920
```

Figure 21.1. Residual Statistics

The PDLREG procedure next prints a table of parameter estimates, standard errors, and *t*-tests, as shown in Figure 21.2.

```
                              The PDLREG Procedure

                                          Standard                 Approx
      Variable        DF      Estimate       Error    t Value    Pr > |t|

      Intercept        1       10.0030      0.0431     231.87      <.0001
      x**0             1        0.4406      0.0378      11.66      <.0001
      x**1             1        0.0113      0.0336       0.34      0.7377
      x**2             1       -0.4108      0.0322     -12.75      <.0001
      x**3             1        0.0331      0.0392       0.84      0.4007
```

Figure 21.2. Parameter Estimates

The preceding table shows the model intercept and the estimated parameters of the lag distribution polynomial. The parameter labeled X**0 is the constant term, α_0, of the distribution polynomial. X**1 is the linear coefficient, α_1, X**2 is the quadratic coefficient, α_2, and X**3 is the cubic coefficient, α_3.

The parameter estimates for the distribution polynomial are not of interest in themselves. Since the PDLREG procedure does not print the orthogonal polynomial basis that it constructs to represent the distribution polynomial, these coefficient values cannot be interpreted.

However, because these estimates are for an orthogonal basis, you can use these results to test the degree of the polynomial. For example, this table shows that the X**3 estimate is not significant; the *p*-value for its *t* ratio is .4007, while the X**2 estimate is highly significant ($p < .0001$). This indicates that a second-degree polynomial may be more appropriate for this data set.

The PDLREG procedure next prints the lag distribution coefficients and a graphical display of these coefficients, as shown in Figure 21.3.

```
                            The PDLREG Procedure

                        Estimate of Lag Distribution

                                            Standard                  Approx
        Variable            Estimate           Error     t Value    Pr > |t|

        x(0)               -0.040150          0.0360       -1.12      0.2677
        x(1)                0.324241          0.0307       10.55      <.0001
        x(2)                0.416661          0.0239       17.45      <.0001
        x(3)                0.289482          0.0315        9.20      <.0001
        x(4)               -0.004926          0.0365       -0.13      0.8929

                       Estimate of Lag Distribution


          Variable        -0.04                                 0.4167

          x(0)            |***|                                      |
          x(1)            |   |*****************************         |
          x(2)            |   |*************************************|
          x(3)            |   |**************************            |
          x(4)            |   |                                      |
```

Figure 21.3. Coefficients and Graph of Estimated Lag Distribution

The lag distribution coefficients are the coefficients of the lagged values of X in the regression model. These coefficients lie on the polynomial curve defined by the parameters shown in Figure 21.2. Note that the estimated values for X(1), X(2), and X(3) are highly significant, while X(0) and X(4) are not significantly different from 0. These estimates are reasonably close to the true values used to generate the simulated data.

The graphical display of the lag distribution coefficients plots the estimated lag distribution polynomial reported in Figure 21.2. The roughly quadratic shape of this plot is another indication that a third-degree distribution curve is not needed for this data set.

Syntax

The following statements can be used with the PDLREG procedure:

PROC PDLREG *option* **;**
  BY *variables* **;**
  MODEL *dependents = effects / options* **;**
  OUTPUT OUT= *SAS-data-set keyword = variables* **;**
  RESTRICT *restrictions* **;**

Functional Summary

The statements and options used with the PDLREG procedure are summarized in the following table:

Description	Statement	Option
Data Set Options		
specify the input data set	PDLREG	DATA=
write predicted values to an output data set	OUTPUT	OUT=
BY-Group Processing		
specify BY-group processing	BY	
Printing Control Options		
request all print options	MODEL	ALL
print correlations of the estimates	MODEL	CORRB
print covariances of the estimates	MODEL	COVB
print DW statistics up to order *j*	MODEL	DW=*j*
print the marginal probability of DW statistics	MODEL	DWPROB
print inverse of the crossproducts matrix	MODEL	I
print details at each iteration step	MODEL	ITPRINT
print Durbin *t* statistic	MODEL	LAGDEP
print Durbin *h* statistic	MODEL	LAGDEP=
suppress printed output	MODEL	NOPRINT
print partial autocorrelations	MODEL	PARTIAL
print standardized parameter estimates	MODEL	STB
print crossproducts matrix	MODEL	XPX
Model Estimation Options		
specify order of autoregressive process	MODEL	NLAG=
suppress intercept parameter	MODEL	NOINT
specify convergence criterion	MODEL	CONVERGE=
specify maximum number of iterations	MODEL	MAXITER=
specify estimation method	MODEL	METHOD=
Output Control Options		
specify confidence limit size	OUTPUT	ALPHACLI=
specify confidence limit size for structural predicted values	OUTPUT	ALPHACLM=
output transformed intercept variable	OUTPUT	CONSTANT=
output lower confidence limit for predicted values	OUTPUT	LCL=
output lower confidence limit for structural predicted values	OUTPUT	LCLM=
output predicted values	OUTPUT	P=
output predicted values of the structural part	OUTPUT	PM=
output residuals from the predicted values	OUTPUT	R=
output residuals from the structural predicted values	OUTPUT	RM=

Description	**Statement**	**Option**
output transformed variables	OUTPUT	TRANSFORM=
output upper confidence limit for the predicted values	OUTPUT	UCL=
output upper confidence limit for the structural predicted values	OUTPUT	UCLM=

PROC PDLREG Statement

PROC PDLREG *option ;*

The PROC PDLREG statement has the following option:

DATA= *SAS-data-set*
specifies the name of the SAS data set containing the input data. If you do not specify the DATA= option, the most recently created SAS data set is used.

In addition, you can place any of the following MODEL statement options in the PROC PDLREG statement, which is equivalent to specifying the option for every MODEL statement: ALL, CONVERGE=, CORRB, COVB, DW=, DWPROB, ITPRINT, MAXITER=, METHOD=, NOINT, NOPRINT, and PARTIAL.

BY Statement

BY *variables ;*

A BY statement can be used with PROC PDLREG to obtain separate analyses on observations in groups defined by the BY variables.

MODEL Statement

MODEL *dependent = effects / options ;*

The MODEL statement specifies the regression model. The keyword MODEL is followed by the dependent variable name, an equal sign, and a list of independent effects. Only one MODEL statement is allowed.

Every variable in the model must be a numeric variable in the input data set. Specify an independent effect with a variable name optionally followed by a polynomial lag distribution specification.

Specifying Independent Effects

The general form of an effect is

variable (length, degree, minimum-degree, constraint)

The term in parentheses following the variable name specifies a polynomial distributed lag (PDL) for the variable. The PDL specification is as follows:

length	specifies the number of lags of the variable to include in the lag distribution.
degree	specifies the maximum degree of the distribution polynomial. If not specified, the degree defaults to the lag length.
minimum-degree	specifies the minimum degree of the polynomial. By default *minimum-degree* is the same as *degree*.
constraint	specifies endpoint restrictions on the polynomial. The value of *constraint* can be FIRST, LAST, or BOTH. If a value is not specified, there are no endpoint restrictions.

If you do not specify the *degree* or *minimum-degree* parameter, but you do specify endpoint restrictions, you must use commas to show which parameter, *degree* or *minimum-degree*, is left out.

MODEL Statement Options

The following options can appear in the MODEL statement after a slash (/):

ALL
prints all the matrices computed during the analysis of the model.

CORRB
prints the matrix of estimated correlations between the parameter estimates.

COVB
prints the matrix of estimated covariances between the parameter estimates.

DW= *j*
prints the generalized Durbin-Watson statistics up to the order of *j*. The default is DW=1. When you specify the LAGDEP or LAGDEP=*name* option, the Durbin-Watson statistic is not printed unless you specify the DW= option.

DWPROB
prints the marginal probability of the Durbin-Watson statistic.

CONVERGE= *value*
sets the convergence criterion. If the maximum absolute value of the change in the autoregressive parameter estimates between iterations is less than this amount, then convergence is assumed. The default is CONVERGE=.001.

I
prints $(\mathbf{X'X})^{-1}$, the inverse of the crossproducts matrix for the model; or, if restrictions are specified, prints $(\mathbf{X'X})^{-1}$ adjusted for the restrictions.

ITPRINT
prints information on each iteration.

LAGDEP
LAGDV
prints the *t* statistic for testing residual autocorrelation when regressors contain lagged dependent variables.

LAGDEP= *name*
LAGDV= *name*
prints the Durbin *h* statistic for testing the presence of first-order autocorrelation when regressors contain the lagged dependent variable whose name is specified as LAGDEP=*name*. When the *h* statistic cannot be computed, the asymptotically equivalent *t* statistic is given.

MAXITER= *number*
sets the maximum number of iterations allowed. The default is MAXITER=50.

METHOD= *value*
specifies the type of estimates for the autoregressive component. The values of the METHOD= option are as follows:

METHOD=ML	specifies the maximum likelihood method
METHOD=ULS	specifies unconditional least squares
METHOD=YW	specifies the Yule-Walker method
METHOD=ITYW	specifies iterative Yule-Walker estimates

The default is METHOD=ML if you specified the LAGDEP or LAGDEP= option; otherwise, METHOD=YW is the default.

NLAG= *m*
NLAG= *(number-list)*
specifies the order of the autoregressive process or the subset of autoregressive lags to be fit. If you do not specify the NLAG= option, PROC PDLREG does not fit an autoregressive model.

NOINT
suppresses the intercept parameter from the model.

NOPRINT
suppresses the printed output.

PARTIAL
prints partial autocorrelations if the NLAG= option is specified.

STB
prints standardized parameter estimates. Sometimes known as a standard partial regression coefficient, a *standardized parameter estimate* is a parameter estimate multiplied by the standard deviation of the associated regressor and divided by the standard deviation of the regressed variable.

XPX
prints the crossproducts matrix, **X’X**, used for the model. **X** refers to the transformed matrix of regressors for the regression.

OUTPUT Statement

OUTPUT *OUT= SAS-data-set keyword=option ... ;*

The OUTPUT statement creates an output SAS data set with variables as specified by the following keyword options. The associated computations for these options are described in the section "Predicted Values" in Chapter 12.

ALPHACLI= *number*
sets the confidence limit size for the estimates of future values of the current realization of the response time series to *number*, where *number* is less than one and greater than zero. The resulting confidence interval has 1-*number* confidence. The default value for *number* is .05, corresponding to a 95% confidence interval.

ALPHACLM= *number*
sets the confidence limit size for the estimates of the structural or regression part of the model to *number*, where *number* is less than one and greater than zero. The resulting confidence interval has 1-*number* confidence. The default value for *number* is .05, corresponding to a 95% confidence interval.

OUT= *SAS-data-set*
names the output data.

The following specifications are of the form *KEYWORD=names*, where *KEYWORD=* specifies the statistic to include in the output data set and *names* gives names to the variables that contain the statistics.

CONSTANT= *variable*
writes the transformed intercept to the output data set.

LCL= *name*
requests that the lower confidence limit for the predicted value (specified in the PREDICTED= option) be added to the output data set under the name given.

LCLM= *name*
requests that the lower confidence limit for the structural predicted value (specified in the PREDICTEDM= option) be added to the output data set under the name given.

PREDICTED= *name*
P=*name*
stores the predicted values in the output data set under the name given.

PREDICTEDM= *name*
PM= *name*
stores the structural predicted values in the output data set under the name given. These values are formed from only the structural part of the model.

RESIDUAL= *name*
R= *name*
stores the residuals from the predicted values based on both the structural and time series parts of the model in the output data set under the name given.

RESIDUALM= *name*
RM= *name*
requests that the residuals from the structural prediction be given.

TRANSFORM= *variables*
requests that the specified variables from the input data set be transformed by the autoregressive model and put in the output data set. If you need to reproduce the data suitable for reestimation, you must also transform an intercept variable. To do this, transform a variable that only takes the value 1 or use the CONSTANT= option.

UCL= *name*
stores the upper confidence limit for the predicted value (specified in the PREDICTED= option) in the output data set under the name given.

UCLM= *name*
stores the upper confidence limit for the structural predicted value (specified in the PREDICTEDM= option) in the output data set under the name given.

For example, the SAS statements

```
proc pdlreg data=a;
   model y=x1 x2;
   output out=b p=yhat r=resid;
```

create an output data set named B. In addition to the input data set variables, the data set B contains the variable YHAT, whose values are predicted values of the dependent variable Y, and RESID, whose values are the residual values of Y.

RESTRICT Statement

RESTRICT *equation , ... , equation ;*

The RESTRICT statement places restrictions on the parameter estimates for covariates in the preceding MODEL statement. A parameter produced by a distributed lag cannot be restricted with the RESTRICT statement.

Each restriction is written as a linear equation. If you specify more than one restriction in a RESTRICT statement, the restrictions are separated by commas.

You can refer to parameters by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model.

RESTRICT statements can be given labels. You can use labels to distinguish results for different restrictions in the printed output. Labels are specified as follows:

label : **RESTRICT** . . . **;**

The following is an example of the use of the RESTRICT statement, in which the coefficients of the regressors X1 and X2 are required to sum to 1.

```
proc pdlreg data=a;
   model y = x1 x2;
   restrict x1 + x2 = 1;
run;
```

Parameter names can be multiplied by constants. When no equal sign appears, the linear combination is set equal to 0. Note that the parameters associated with the variables are restricted, not the variables themselves. Here are some examples of valid RESTRICT statements:

```
restrict x1 + x2 = 1;
restrict x1 + x2 - 1;
restrict 2 * x1 = x2 + x3 , intercept + x4 = 0;
restrict x1 = x2 = x3 = 1;
restrict 2 * x1 - x2;
```

Restricted parameter estimates are computed by introducing a Lagrangian parameter λ for each restriction (Pringle and Raynor 1971). The estimates of these Lagrangian parameters are printed in the parameter estimates table. If a restriction cannot be applied, its parameter value and degrees of freedom are listed as 0.

The Lagrangian parameter, λ, measures the sensitivity of the SSE to the restriction. If the restriction is changed by a small amount ϵ, the SSE is changed by $2\lambda\epsilon$.

The *t* ratio tests the significance of the restrictions. If λ is zero, the restricted estimates are the same as the unrestricted ones.

You can specify any number of restrictions on a RESTRICT statement, and you can use any number of RESTRICT statements. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

Details

Missing Values

The PDLREG procedure skips any observations at the beginning of the data set that have missing values. The procedure uses all observations with nonmissing values for all the independent and dependent variables such that the lag distribution has sufficient nonmissing lagged independent variables.

Polynomial Distributed Lag Estimation

The simple finite distributed lag model is expressed in the form

$$y_t = \alpha + \sum_{i=0}^{p} \beta_i x_{t-i} + \epsilon_t$$

When the lag length (*p*) is long, severe multicollinearity can occur. Use the Almon or *polynomial distributed lag* model to avoid this problem, since the relatively low degree *d* ($\leq p$) polynomials can capture the true lag distribution. The lag coefficient can be written in the Almon polynomial lag

$$\beta_i = \alpha_0^* + \sum_{j=1}^{d} \alpha_j^* i^j$$

Emerson (1968) proposed an efficient method of constructing orthogonal polynomials from the preceding polynomial equation as

$$\beta_i = \alpha_0 + \sum_{j=1}^{d} \alpha_j f_j(i)$$

where $f_j(i)$ is a polynomial of degree *j* in the lag length *i*. The polynomials $f_j(i)$ are chosen so that they are orthogonal:

$$\sum_{i=1}^{n} w_i f_j(i) f_k(i) = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

where w_i is the weighting factor, and $n = p + 1$. PROC PDLREG uses the equal weights ($w_i = 1$) for all *i*. To construct the orthogonal polynomials, the following recursive relation is used:

$$f_j(i) = (A_j i + B_j) f_{j-1}(i) - C_j f_{j-2}(i) \qquad j = 1, \ldots, d$$

The constants A_j, B_j, and C_j are determined as follows:

$$\begin{aligned}
A_j &= \left\{ \sum_{i=1}^{n} w_i i^2 f_{j-1}^2(i) \quad - \left(\sum_{i=1}^{n} w_i i f_{j-1}^2(i) \right)^2 \right. \\
&\qquad \left. - \left(\sum_{i=1}^{n} w_i i f_{j-1}(i) f_{j-2}(i) \right)^2 \right\}^{-1/2} \\
B_j &= -A_j \sum_{i=1}^{n} w_i i f_{j-1}^2(i) \\
C_j &= A_j \sum_{i=1}^{n} w_i i f_{j-1}(i) f_{j-2}(i)
\end{aligned}$$

where $f_{-1}(i) = 0$ and $f_0(i) = 1/\sqrt{\sum_{i=1}^{n} w_i}$.

PROC PDLREG estimates the orthogonal polynomial coefficients, $\alpha_0, \ldots, \alpha_d$, to compute the coefficient estimate of each independent variable (X) with distributed lags. For example, if an independent variable is specified as X(9,3), a third-degree polynomial is used to specify the distributed lag coefficients. The third-degree polynomial is fit as a constant term, a linear term, a quadratic term, and a cubic term. The four terms are constructed to be orthogonal. In the output produced by the PDLREG procedure for this case, parameter estimates with names X**0, X**1, X**2, and X**3 correspond to $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2$, and $\hat{\alpha}_3$, respectively. A test using the *t* statistic and the approximate *p*-value ("Approx Pr $>$ $|t|$") associated with X**3 can determine whether a second-degree polynomial rather than a third-degree polynomial is appropriate. The estimates of the ten lag coefficients associated with the specification X(9,3) are labeled X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7), X(8), and X(9).

Autoregressive Error Model Estimation

The PDLREG procedure uses the same autoregressive error model estimation methods as the AUTOREG procedure. These two procedures share the same computational resources for computing estimates. See Chapter 12 for details about estimation methods for autoregressive error models.

OUT= Data Set

The OUT= data set produced by the PDLREG procedure's OUTPUT statement is similar in form to the OUT= data set produced by the AUTOREG procedure. See Chapter 12 for details on the OUT= data set.

Printed Output

The PDLREG procedure prints the following items:

1. the name of the dependent variable
2. the ordinary least squares (OLS) estimates
3. the estimates of autocorrelations and of the autocovariance, and if line size permits, a graph of the autocorrelation at each lag. The autocorrelation for lag 0 is 1. These items are printed if you specify the NLAG= option.
4. the partial autocorrelations if the PARTIAL and NLAG= options are specified. The first partial autocorrelation is the autocorrelation for lag 1.
5. the preliminary mean square error, which results from solving the Yule-Walker equations if you specify the NLAG= option
6. the estimates of the autoregressive parameters, their standard errors, and the ratios of estimates to standard errors (*t*) if you specify the NLAG= option
7. the statistics of fit for the final model if you specify the NLAG= option. These include the error sum of squares (SSE), the degrees of freedom for error (DFE), the mean square error (MSE), the root mean square error (Root MSE), the Schwarz information criterion (SBC), the Akaike's information criterion

(AIC), the regression R^2 (Regress R-Square), the total R^2 (Total R-Square), and the Durbin-Watson statistic (Durbin-Watson). See Chapter 12 for details of the regression R^2 and the total R^2.

8. the parameter estimates for the structural model (B), a standard error estimate, the ratio of estimate to standard error (t), and an approximation to the significance probability for the parameter being 0 ("Approx Pr $> |t|$")
9. a plot of the lag distribution (estimate of lag distribution)
10. the covariance matrix of the parameter estimates if the COVB option is specified

ODS Table Names

PROC PDLREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 21.1. ODS Tables Produced in PROC PDLREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ARParameterEstimates	Estimates of Autoregressive Parameters	NLAG=
CholeskyFactor	Cholesky Root of Gamma	
Coefficients	Coefficients for First NLAG Observations	NLAG=
ConvergenceStatus	Convergence Status table	default
CorrB	Correlation of Parameter Estimates	CORRB
CorrGraph	Estimates of Autocorrelations	NLAG=
CovB	Covariance of Parameter Estimates	COVB
DependenceEquations	Linear dependence equation	
Dependent	Dependent variable	default
DWTest	Durbin-Watson Statistics	DW=
ExpAutocorr	Expected Autocorrelations	NLAG=
FitSummary	Summary of regression	default
GammaInverse	Gamma Inverse	
IterHistory	Iteration History	ITPRINT
LagDist	Lag Distribution	ALL
ParameterEstimates	Parameter Estimates	default
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=
PartialAutoCorr	Partial autocorrelation	PARTIAL
PreMSE	Preliminary MSE	NLAG=
XPXIMatrix	Inverse X'X Matrix	XPX
XPXMatrix	X'X Matrix	XPX

Table 21.1. (continued)

ODS Table Name	Description	Option
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction table	default

Examples

Example 21.1. Industrial Conference Board Data

In the following example, a second-degree Almon polynomial lag model is fit to a model with a five-period lag, and dummy variables are used for quarter effects. The PDL model is estimated using capital appropriations data series for the period 1952 to 1967. The estimation model is written

$$\begin{aligned} CE_t = {} & a_0 + b_1 Q1_t + b_2 Q2_t + b_3 Q3_t \\ & + c_0 CA_t + c_1 CA_{t-1} + \ldots + c_5 CA_{t-5} \end{aligned}$$

where CE represents capital expenditures and CA represents capital appropriations.

```
title 'National Industrial Conference Board Data';
title2 'Quarterly Series - 1952Q1 to 1967Q4';

data a;
   input ce ca @@;
   qtr = mod( _n_-1, 4 ) + 1;
   q1  = qtr=1;
   q2  = qtr=2;
   q3  = qtr=3;
cards;
   2072 1660 2077 1926 2078 2181 2043 1897 2062 1695
   2067 1705 1964 1731 1981 2151 1914 2556 1991 3152
   2129 3763 2309 3903 2614 3912 2896 3571 3058 3199
   3309 3262 3446 3476 3466 2993 3435 2262 3183 2011
   2697 1511 2338 1631 2140 1990 2012 1993 2071 2520
   2192 2804 2240 2919 2421 3024 2639 2725 2733 2321
   2721 2131 2640 2552 2513 2234 2448 2282 2429 2533
   2516 2517 2534 2772 2494 2380 2596 2568 2572 2944
   2601 2629 2648 3133 2840 3449 2937 3764 3136 3983
   3299 4381 3514 4786 3815 4094 4093 4870 4262 5344
   4531 5433 4825 5911 5160 6109 5319 6542 5574 5785
   5749 5707 5715 5412 5637 5465 5383 5550 5467 5465
;

proc pdlreg data=a;
```

```
      model ce = q1 q2 q3 ca(5,2) / dwprob;
   run;
```

The printed output produced by the PDLREG procedure is shown in Output 21.1.1. The small Durbin-Watson test indicates autoregressive errors.

Output 21.1.1. Printed Output Produced by PROC PDLREG

```
                   National Industrial Conference Board Data
                      Quarterly Series - 1952Q1 to 1967Q4

                              The PDLREG Procedure

                            Dependent Variable    ce


                        Ordinary Least Squares Estimates

         SSE                    1205186.4    DFE                      48
         MSE                        25108    Root MSE          158.45520
         SBC                    733.84921    AIC              719.797878
         Regress R-Square          0.9834    Total R-Square       0.9834
         Durbin-Watson             0.6157    Pr < DW              <.0001
         Pr > DW                   1.0000


                                            Standard                 Approx
      Variable        DF     Estimate          Error    t Value    Pr > |t|

      Intercept        1     210.0109        73.2524       2.87      0.0061
      q1               1     -10.5515        61.0634      -0.17      0.8635
      q2               1     -20.9887        59.9386      -0.35      0.7277
      q3               1     -30.4337        59.9004      -0.51      0.6137
      ca**0            1       0.3760       0.007318      51.38      <.0001
      ca**1            1       0.1297         0.0251       5.16      <.0001
      ca**2            1       0.0247         0.0593       0.42      0.6794


                          Estimate of Lag Distribution

                                           Standard                 Approx
        Variable           Estimate           Error    t Value    Pr > |t|

        ca(0)              0.089467          0.0360       2.49      0.0165
        ca(1)              0.104317          0.0109       9.56      <.0001
        ca(2)              0.127237          0.0255       5.00      <.0001
        ca(3)              0.158230          0.0254       6.24      <.0001
        ca(4)              0.197294          0.0112      17.69      <.0001
        ca(5)              0.244429          0.0370       6.60      <.0001
                         Estimate of Lag Distribution


          Variable        0                                   0.2444

          ca(0)           |***************                         |
          ca(1)           |*****************                       |
          ca(2)           |*********************                   |
          ca(3)           |***************************             |
          ca(4)           |*********************************       |
          ca(5)           |*****************************************|
```

The following statements use the REG procedure to fit the same polynomial distributed lag model. A DATA step computes lagged values of the regressor X, and

RESTRICT statements are used to impose the polynomial lag distribution. Refer to Judge, Griffiths, Hill, Lutkepohl, and Lee (1985, pp 357–359) for the restricted least squares estimation of the Almon distributed lag model.

```
   data b;
      set a;
      ca_1 = lag( ca );
      ca_2 = lag2( ca );
      ca_3 = lag3( ca );
      ca_4 = lag4( ca );
      ca_5 = lag5( ca );
   run;

   proc reg data=b;
      model  ce = q1 q2 q3 ca ca_1 ca_2 ca_3 ca_4 ca_5;
      restrict   - ca + 5*ca_1 - 10*ca_2 + 10*ca_3 - 5*ca_4 +   ca_5;
      restrict     ca - 3*ca_1 +  2*ca_2 +  2*ca_3 - 3*ca_4 +   ca_5;
      restrict  -5*ca + 7*ca_1 +  4*ca_2 -  4*ca_3 - 7*ca_4 + 5*ca_5;
   run;
```

The REG procedure output is shown in Output 21.1.2.

Output 21.1.2. Printed Output Produced by PROC REG

```
                              The REG Procedure
                                Model: MODEL1
                           Dependent Variable: ce

                             Analysis of Variance

                                     Sum of           Mean
Source                   DF         Squares         Square    F Value    Pr > F

Model                     6        71343377       11890563     473.58    <.0001
Error                    48         1205186          25108
Corrected Total          54        72548564


             Root MSE            158.45520    R-Square     0.9834
             Dependent Mean     3185.69091    Adj R-Sq     0.9813
             Coeff Var             4.97397


                             Parameter Estimates

                            Parameter       Standard
     Variable      DF        Estimate          Error    t Value    Pr > |t|

     Intercept      1       210.01094       73.25236       2.87      0.0061
     q1             1       -10.55151       61.06341      -0.17      0.8635
     q2             1       -20.98869       59.93860      -0.35      0.7277
     q3             1       -30.43374       59.90045      -0.51      0.6137
     ca             1         0.08947        0.03599       2.49      0.0165
     ca_1           1         0.10432        0.01091       9.56      <.0001
     ca_2           1         0.12724        0.02547       5.00      <.0001
     ca_3           1         0.15823        0.02537       6.24      <.0001
     ca_4           1         0.19729        0.01115      17.69      <.0001
     ca_5           1         0.24443        0.03704       6.60      <.0001
     RESTRICT      -1       623.63242          12697       0.05      0.9614*
     RESTRICT      -1           18933          44803       0.42      0.6772*
     RESTRICT      -1           10303          18422       0.56      0.5814*

                * Probability computed using beta distribution.
```

Example 21.2. Money Demand Model

This example estimates the demand for money using the following dynamic specification:

$$m_t = a_0 + b_0 m_{t-1} + \sum_{i=0}^{5} c_i y_{t-i} + \sum_{i=0}^{2} d_i r_{t-i} + \sum_{i=0}^{3} f_i p_{t-i} + u_t$$

where

m_t =log of real money stock (M1)

y_t =log of real GNP

r_t =interest rate (commercial paper rate)

p_t =inflation rate

c_i, d_i, and f_i $(i > 0)$are coefficients for the lagged variables

The following DATA step reads the data and transforms the real money and real GNP variables using the natural logarithm. Refer to Balke and Gordon (1986) for a description of the data.

```
data a;
   input m1 gnp gdf r @@;
   m    = log( 100 * m1 / gdf );
   lagm = lag( m );
   y    = log( gnp );
   p    = log( gdf / lag( gdf ) );
   date = intnx( 'qtr', '1jan1968'd, _n_-1 );
   format date yyqc6.;
   label m    = 'Real Money Stock (M1)'
         lagm = 'Lagged Real Money Stock'
         y    = 'Real GNP'
         r    = 'Commercial Paper Rate'
         p    = 'Inflation Rate';
cards;
   ... data lines are omitted ...
;

proc print data=a(obs=5);
   var date m lagm y r p;
run;
```

Output 21.2.1 shows a partial list of the data set.

Output 21.2.1. Partial List of the Data Set A

```
Obs     date        m        lagm        y        r        p

  1    1968:1    5.44041     .        6.94333    5.58     .
  2    1968:2    5.44732  5.44041     6.96226    6.08    0.011513
  3    1968:3    5.45815  5.44732     6.97422    5.96    0.008246
  4    1968:4    5.46492  5.45815     6.97661    5.96    0.014865
  5    1969:1    5.46980  5.46492     6.98855    6.66    0.011005
```

The regression model is written for the PDLREG procedure with a MODEL statement. The LAGDEP= option is specified to test for the serial correlation in disturbances since regressors contain the lagged dependent variable LAGM.

```
title 'Money Demand Estimation using Distributed Lag Model';
title2 'Quarterly Data - 1968Q2 to 1983Q4';

proc pdlreg data=a;
   model m = lagm y(5,3) r(2, , ,first) p(3,2) / lagdep=lagm;
run;
```

The estimated model is shown in Output 21.2.2 and Output 21.2.3.

Output 21.2.2. Parameter Estimates

```
              Money Demand Estimation using Distributed Lag Model
                       Quarterly Data - 1968Q2 to 1983Q4

                              The PDLREG Procedure

                  Dependent Variable                          m
                                              Real Money Stock (M1)


                         Ordinary Least Squares Estimates

         SSE                  0.00169815    DFE                       48
         MSE                   0.0000354    Root MSE             0.00595
         SBC                  -404.60169    AIC                -427.4546
         Regress R-Square         0.9712    Total R-Square        0.9712
         Durbin h                -0.7533    Pr < h                0.2256


                                             Standard                 Approx
      Variable        DF      Estimate          Error    t Value    Pr > |t|

      Intercept        1       -0.1407         0.2625      -0.54      0.5943
      lagm             1        0.9875         0.0425      23.21      <.0001
      y**0             1        0.0132       0.004531       2.91      0.0055
      y**1             1       -0.0704         0.0528      -1.33      0.1891
      y**2             1        0.1261         0.0786       1.60      0.1154
      y**3             1       -0.4089         0.1265      -3.23      0.0022
      r**0             1     -0.000186       0.000336      -0.55      0.5816
      r**1             1      0.002200       0.000774       2.84      0.0065
      r**2             1      0.000788       0.000249       3.16      0.0027
      p**0             1       -0.6602         0.1132      -5.83      <.0001
      p**1             1        0.4036         0.2321       1.74      0.0885
      p**2             1       -1.0064         0.2288      -4.40      <.0001


                                             Standard                 Approx
      Restriction     DF       L Value          Error    t Value    Pr > |t|

      r(-1)           -1        0.0164       0.007275       2.26      0.0223
```

Output 21.2.3. Estimates for Lagged Variables

```
                 Money Demand Estimation using Distributed Lag Model
                          Quarterly Data - 1968Q2 to 1983Q4

                                 The PDLREG Procedure

                             Estimate of Lag Distribution

                                            Standard                  Approx
        Variable            Estimate           Error     t Value    Pr > |t|

        y(0)                0.268619          0.0910        2.95      0.0049
        y(1)               -0.196484          0.0612       -3.21      0.0024
        y(2)               -0.163148          0.0537       -3.04      0.0038
        y(3)                0.063850          0.0451        1.42      0.1632
        y(4)                0.179733          0.0588        3.06      0.0036
        y(5)               -0.120276          0.0679       -1.77      0.0827

                            Estimate of Lag Distribution


          Variable        -0.196           0                   0.2686

          y(0)            |                |************************|
          y(1)            |****************|                        |
          y(2)            |   *************|                        |
          y(3)            |                |******                  |
          y(4)            |                |****************        |
          y(5)            |       *********|                        |
      Money Demand Estimation using Distributed Lag Model
               Quarterly Data - 1968Q2 to 1983Q4

                      The PDLREG Procedure

                  Estimate of Lag Distribution

                                 Standard                Approx
Variable            Estimate        Error    t Value    Pr > |t|

r(0)               -0.001341     0.000388      -3.45      0.0012
r(1)               -0.000751     0.000234      -3.22      0.0023
r(2)                0.001770     0.000754       2.35      0.0230

                 Estimate of Lag Distribution


  Variable        -0.001            0                  0.0018

  r(0)            |*****************|                       |
  r(1)            |         *********|                       |
  r(2)            |                 |***********************|


                  Estimate of Lag Distribution

                                 Standard                Approx
Variable            Estimate        Error    t Value    Pr > |t|

p(0)               -1.104051       0.2027      -5.45      <.0001
p(1)                0.082892       0.1257       0.66      0.5128
p(2)                0.263391       0.1381       1.91      0.0624
p(3)               -0.562556       0.2076      -2.71      0.0093

                 Estimate of Lag Distribution


  Variable        -1.104                           0   0.2634

  p(0)            |********************************|       |
  p(1)            |                                |***    |
  p(2)            |                                |********|
  p(3)            |                ****************|       |
```

References

Balke, N.S. and Gordon, R.J. (1986), "Historical Data," in *The American Business Cycle*, ed. R.J. Gordon, Chicago: The University of Chicago Press.

Emerson, P.L. (1968), "Numerical Construction of Orthogonal Polynomials from a General Recurrence Formula," *Biometrics*, 24, 695–701.

Gallant, A.R. and Goebel, J.J. (1976), "Nonlinear Regression with Autoregressive Errors," *Journal of the American Statistical Association*, 71, 961–967.

Harvey, A.C. (1981), *The Econometric Analysis of Time Series*, New York: John Wiley & Sons, Inc.

Johnston, J. (1972), *Econometric Methods*, Second Edition, New York: McGraw-Hill Book Co.

Judge, G.G., Griffiths, W.E., Hill, R.C., Lutkepohl, H., and Lee, T.C. (1985), *The Theory and Practice of Econometrics*, Second Edition, New York: John Wiley & Sons, Inc.

Park, R.E. and Mitchell, B.M. (1980), "Estimating the Autocorrelated Error Model with Trended Data," *Journal of Econometrics*, 13, 185–201.

Pringle, R.M. and Raynor, A.A. (1971), *Generalized Inverse Matrices with Applications to Statistics*, New York: Hafner Publishing Company.

Chapter 22
The QLIM Procedure

Chapter Contents

Chapter 22
The QLIM Procedure

Overview

The QLIM (Qualitative and LImited dependent variable Model) procedure analyzes univariate and multivariate limited dependent variable models where dependent variables take discrete values or dependent variables are observed only in a limited range of values. This procedure includes logit, probit, tobit, selection, and multivariate models. The multivariate model can contain discrete choice and limited endogenous variables as well as continuous endogenous variables.

The QLIM procedure supports the following models:

- linear regression model with heteroscedasticity
- probit with heteroscedasticity
- logit with heteroscedasticity
- tobit (censored and truncated) with heteroscedasticity
- Box-Cox regression with heteroscedasticity
- bivariate probit
- bivariate tobit
- sample selection and switching regression models
- multivariate limited dependent variables

Getting Started

The QLIM procedure is similar in use to the other regression or simultaneous equations model procedures in the SAS System. For example, the following statements are used to estimate a binary choice model using the probit probability function:

```
proc qlim data=a;
   model y = x1;
   endogenous y ~ discrete;
run;
```

The response variable, y, is numeric and has discrete values. PROC QLIM enables the user to specify the type of endogenous variables in the ENDOGENOUS statement. The binary probit model can be also specified as follows:

```
model y = x1 / discrete;
```

When multiple endogenous variables are specified in the QLIM procedure, these equations are estimated as a system. Multiple endogenous variables can be specified with one MODEL statement in the QLIM procedure when these models have the same exogenous variables:

```
model y1 y2 = x1 x2 / discrete;
```

The preceding specification is equivalent to

```
proc qlim data=a;
   model y1 = x1 x2;
   model y2 = x1 x2;
   endogenous y1 y2 ~ discrete;
run;
```

The standard tobit model is estimated by specifying the endogenous variable to be truncated or censored. The limits of the dependent variable can be specified with the CENSORED or TRUNCATED option in the ENDOGENOUS or MODEL statement when the data are limited by specific values or variables. For example, the two-limit censored model requires two variables that contain the lower (bottom) and upper (top) bound.

```
proc qlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=bottom ub=top);
run;
```

The bounds can be numbers if they are fixed for all observations in the data set. For example, the standard tobit model can be specified as

```
proc qlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
run;
```

Introductory Example: Binary Probit and Logit Models

The following example illustrates the use of PROC QLIM. The data are originally published by Mroz (1987) and downloaded from Wooldridge (2002). This data set is based on a sample of 753 married white women. The dependent variable is a discrete variable of labor force participation (inlf). Explanatory variables are the number of children ages 5 or younger (kidslt6), the number of children ages 6 to 18 (kidsge6), the woman's age (age), the woman's years of schooling (educ), wife's labor experience (exper), square of experience (expersq), and the family income excluding the wife's wage (nwifeinc). The program (with data values omitted) is illustrated below.

```
title1 "Binary Data";
data mroz;
 input inlf nwifeinc educ exper expersq age kidslt6 kidsge6 lwage;
datalines;
1 10.91006    12 14  196   32  1   0   1.210154

  ...

;
run;


proc qlim data=mroz;
   model inlf = nwifeinc educ exper expersq age kidslt6 kidsge6  / discrete;
run;
```

Results of this analysis are shown in the following four figures. In the first table, shown in Figure 22.1, PROC QLIM provides frequency information on each choice. In this example, 428 women participate in the labor force (inlf=1).

```
                              Binary Data

                          The QLIM Procedure

                    Discrete Response Profile of inlf

              Index          Value           Frequency    Percent

                1              0                   325      43.16
                2              1                   428      56.84
```

Figure 22.1. Choice Frequency Summary

The second table is the estimation summary table shown in Figure 22.2. Included are the number of dependent variables, names of dependent variables, the number of observations, the log-likelihood function value, the maximum absolute gradient, the number of iterations, AIC, and Schwarz criterion.

```
                          Model Fit Summary

                   Number of Endogenous Variables             1
                   Endogenous Variable                     inlf
                   Number of Observations                   753
                   Log Likelihood                    -401.30219
                   Maximum Absolute Gradient          0.0004984
                   Number of Iterations                      15
                   AIC                                818.60439
                   Schwarz Criterion                  855.59691
```

Figure 22.2. Fit Summary Table of Binary Probit

Goodness-of-fit measures are displayed in Figure 22.3. All measures except McKelvey-Zavoina's definition are based on the log-likelihood function value. The likelihood ratio test statistic has chi-square distribution conditional on the null hypothesis that all slope coefficients are zero. In this example, the likelihood ratio statistic is used to test the hypothesis that kidslt6=kidge6=age=educ=exper=expersq=nwifeinc= 0.

```
                              Goodness-of-Fit Measures

  Measure                        Value    Formula

  Likelihood Ratio (R)          227.14    2 * (LogL - LogL0)
  Upper Bound of R (U)          1029.7    - 2 * LogL0
  Aldrich-Nelson                0.2317    R / (R+N)
  Cragg-Uhler 1                 0.2604    1 - exp(-R/N)
  Cragg-Uhler 2                 0.3494    (1-exp(-R/N)) / (1-exp(-U/N))
  Estrella                      0.2888    1 - (1-R/U)^(U/N)
  Adjusted Estrella             0.2693    1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
  McFadden's LRI                0.2206    R / U
  Veall-Zimmermann              0.4012    (R * (U+N)) / (U * (R+N))
  McKelvey-Zavoina              0.4025

  N = # of observations, K = # of regressors
```

Figure 22.3. Goodness of Fit

Finally, the parameter estimates and standard errors are shown in Figure 22.4.

```
                               Parameter Estimates

                                           Standard                 Approx
       Parameter         Estimate             Error    t Value    Pr > |t|

       Intercept         0.270077          0.508594       0.53      0.5954
       nwifeinc         -0.012024          0.004840      -2.48      0.0130
       educ              0.130905          0.025254       5.18      <.0001
       exper             0.123348          0.018716       6.59      <.0001
       expersq          -0.001887          0.000600      -3.15      0.0017
       age              -0.052853          0.008477      -6.23      <.0001
       kidslt6          -0.868329          0.118522      -7.33      <.0001
       kidsge6           0.036005          0.043477       0.83      0.4076
```

Figure 22.4. Parameter Estimates of Binary Probit

When the error term has a logistic distribution, the binary logit model is estimated. To specify a logistic distribution, add D=LOGIT option as follows,

```
   proc qlim data=mroz;
     model inlf = nwifeinc educ exper expersq age kidslt6 kidsge6  / discrete(d=logit);
   run;
```

The estimated parameters are shown in Figure 22.5.

```
                              The QLIM Procedure

                             Parameter Estimates

                                            Standard                 Approx
         Parameter         Estimate            Error    t Value    Pr > |t|

         Intercept         0.425452         0.860371       0.49      0.6210
         nwifeinc         -0.021345         0.008421      -2.53      0.0113
         educ              0.221170         0.043440       5.09      <.0001
         exper             0.205870         0.032057       6.42      <.0001
         expersq          -0.003154         0.001016      -3.10      0.0019
         age              -0.088024         0.014573      -6.04      <.0001
         kidslt6          -1.443354         0.203585      -7.09      <.0001
         kidsge6           0.060112         0.074790       0.80      0.4215
```

Figure 22.5. Parameter Estimates of Binary Logit

The heteroscedastic logit model can be estimated using the HETERO statement. If the variance of the logit model is a function of the family income level excluding wife's income (nwifeinc), the variance can be specified as

$$Var(\epsilon_i) = \sigma^2 \exp(\gamma \, \text{nwifeinc}_i)$$

where σ^2 is normalized to 1 because the dependent variable is discrete. The following SAS statements estimate the heteroscedastic logit model:

```
proc qlim data=mroz;
  model inlf = nwifeinc educ exper expersq age kidslt6 kidsge6 / discrete(d=logit);
  hetero inlf ~ nwifeinc / noconst;
run;
```

The parameter estimate (γ) of the heteroscedasticity variable is listed as _H.nwifeinc; see Figure 22.6.

```
                              The QLIM Procedure

                             Parameter Estimates

                                             Standard                 Approx
        Parameter          Estimate             Error    t Value    Pr > |t|

        Intercept          0.510444          0.983616       0.52      0.6038
        nwifeinc          -0.026778          0.012115      -2.21      0.0271
        educ               0.255547          0.061765       4.14      <.0001
        exper              0.234105          0.046736       5.01      <.0001
        expersq           -0.003613          0.001237      -2.92      0.0035
        age               -0.100878          0.021524      -4.69      <.0001
        kidslt6           -1.645206          0.311737      -5.28      <.0001
        kidsge6            0.066941          0.085630       0.78      0.4344
        _H.nwifeinc        0.013280          0.013636       0.97      0.3301
```

Figure 22.6. Parameter Estimates of Binary Logit with Heteroscedasticity

Syntax

The QLIM procedure is controlled by the following statements:

PROC QLIM *options* **;**
  BOUNDS *bound1 [, bound2 . . .]* **;**
  BY *variables* **;**
  CLASS *variables* **;**
  ENDOGENOUS *variables* ∼ *options* **;**
  HETERO *dependent variables* ∼ *exogenous variables / options* **;**
  INIT *initvalue1 [, initvalue2 . . .]* **;**
  MODEL *dependent variables = regressors / options* **;**
  NLOPTIONS *options* **;**
  OUTPUT *options* **;**
  RESTRICT *restriction1 [, restriction2 . . .]* **;**
  WEIGHT *variable* **;**

At least one MODEL statement is required. If more than one MODEL statement is used, the QLIM procedure estimates a system of models. Main effects and higher-order terms can be specified in the MODEL statement, similar to the GLM procedure, and the PROBIT procedure in SAS/STAT. If a CLASS statement is used, it must precede the MODEL statement.

Functional Summary

The statements and options used with the QLIM procedure are summarized in the following table:

Description	Statement	Option
Data Set Options		
specify the input data set	QLIM	DATA=
write parameter estimates to an output data set	QLIM	OUTEST=
write predictions to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
specify BY-group processing	BY	
specify classification variables	CLASS	
specify a weight variable	WEIGHT	
Printing Control Options		
request all printing options	QLIM	PRINTALL
print correlation matrix of the estimates	QLIM	CORRB
print covariance matrix of the estimates	QLIM	COVB
print a summary iteration listing	QLIM	ITPRINT

Description	Statement	Option
suppress the normal printed output	QLIM	NOPRINT
Options to Control the Optimization Process		
specify the optimization method	QLIM	METHOD=
specify the optimization options	NLOPTIONS	see Chapter 10
set initial values for parameters	INIT	
set linear restrictions on parameters	BOUNDS	
	RESTRICT	
Model Estimation Options		
specify options specific to Box-Cox transformation	MODEL	BOXCOX()
suppress the intercept parameter	MODEL	NOINT
specify a seed for pseudo-random number generation	QLIM	SEED=
specify number of draws for Monte Carlo integration	QLIM	NDRAW=
specify method to calculate parameter covariance	QLIM	COVEST=
Endogenous Variable Options		
specify discrete variable	ENDOGENOUS	DISCRETE()
specify censored variable	ENDOGENOUS	CENSORED()
specify truncated variable	ENDOGENOUS	TRUNCATED()
specify variable selection condition	ENDOGENOUS	SELECT()
Heteroscedasticity Model Options		
specify the function for heteroscedasticity models	HETERO	LINK=
square the function for heteroscedasticity models	HETERO	SQUARE
specify no constant for heteroscedasticity models	HETERO	NOCONST
Output Control Options		
output predicted values	OUTPUT	PREDICTED
output structured part	OUTPUT	XBETA
output residuals	OUTPUT	RESIDUAL
output error standard deviation	OUTPUT	ERRSTD
output marginal effects	OUTPUT	MARGINAL
output probability for the current response	OUTPUT	PROB
output probability for all responses	OUTPUT	PROBALL
output expected value	OUTPUT	EXPECTED
output conditional expected value	OUTPUT	CONDITIONAL
output inverse Mills ratio	OUTPUT	MILLS
include covariances in the OUTEST= data set	QLIM	COVOUT
include correlations in the OUTEST= data set	QLIM	CORROUT

PROC QLIM Statement

PROC QLIM *options* **;**

The following options can be used in the PROC QLIM statement:

Data Set Options

DATA= SAS-data-set
specifies the input SAS data set. If the DATA= option is not specified, PROC QLIM uses the most recently created SAS data set.

Output Data Set Options

OUTEST= SAS-data-set
writes the parameter estimates to an output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

NOPRINT
suppresses the normal printed output but does not suppress error listings. If NOPRINT option is set, then any other print option is turned off.

PRINTALL
turns on all the printing-control options. The options set by PRINTALL are COVB, CORRB.

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

ITPRINT
prints the initial parameter estimates, convergence criteria, all constraints of the optimization. At each iteration, objective function value, step size, maximum gradient and slope of search direction are printed as well.

Model Estimation Options

COVEST= covariance-option
specify method to calculate the covariance matrix of parameter estimates. The supported covariance types are

OP	specifies the covariance from the outer product matrix

HESSIAN	specifies the covariance from the inverse Hessian matrix
QML	specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates)

The default is COVEST=HESSIAN.

NDRAW= value
specify number of draws for Monte Carlo integration.

SEED= value
specify a seed for pseudo-random number generation in Monte Carlo integration.

Options to Control the Optimization Process

PROC QLIM uses the NonLinear Optimization (NLO) subsystem to perform nonlinear optimization tasks. All the NLO options are available from the NLOPTIONS statement. For details, see Chapter 10, "Nonlinear Optimization Methods."

METHOD=*value*

specifies the optimization method. If this option is specified, it overwrites the TECH= option in NLOPTIONS statement. Valid values are

CONGRA	performs a conjugate-gradient optimization
DBLDOG	performs a version of double dogleg optimization
NMSIMP	performs a Nelder-Mead simplex optimization
NEWRAP	performs a Newton-Raphson optimization combining a line-search algorithm with ridging
NRRIDG	performs a Newton-Raphson optimization with ridging
QUANEW	performs a quasi-Newton optimization
TRUREG	performs a trust region optimization

The default method is QUANEW.

BOUNDS Statement

BOUNDS *bound1 [, bound2 ...]* **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the QLIM procedure. Any number of BOUNDS statements can be specified.

Each *bound* is composed of variables and constants and inequality operators:

item operator item [operator item [operator item . . .]]

Each *item* is a constant, the name of a parameter, or a list of parameter names. See the "Parameter Names" section for more details on how parameters are named in the QLIM procedure. Each *operator* is '<', '>', '<=', or '>='.

Both the BOUNDS statement and the RESTRICT statement can be used to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See the "RESTRICT Statement" section on page 1334 for more information.

The following BOUNDS statement constrains the estimates of the parameters associated with the variable ttime and the variables x1 through x10 to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < ttime x1-x10 < 1;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC QLIM to obtain separate analyses on observations in groups defined by the BY variables.

CLASS Statement

CLASS *variables* **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. See the discussion of the FORMAT procedure in *SAS Language Reference: Dictionary*.

If the CLASS statement is used, it must appear before any of the MODEL statements.

ENDOGENOUS Statement

ENDOGENOUS *variables* ~ *options* **;**

The ENDOGENOUS statement specifies the type of endogenous variables.

Discrete Variable Options

DISCRETE <(*discrete-options***) >**
specifies that the endogenous variables in this statement are discrete. Valid *discrete-options* are as follows:

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sorting order for the levels of the discrete variables specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. The following table shows how PROC QLIM interprets values of the ORDER= option.

Value of ORDER=	**Levels Sorted By**
DATA	order of appearance in the input data set
FORMATTED	formatted value
FREQ	descending frequency count; levels with the most observations come first in the order
INTERNAL	unformatted value

By default, ORDER=FORMATTED. For the values FORMATTED and INTERNAL, the sort order is machine dependent. For more information on sorting order, see the chapter on the SORT procedure in the *SAS Procedures Guide*.

DISTRIBUTION=*distribution-type*
DIST=*distribution-type*
D=*distribution-type*
specifies the cumulative distribution function used to model the response probabilities. Valid values for *distribution-type* are

NORMAL	the normal distribution for the probit model
LOGISTIC	the logistic distribution for the logit model

By default, DISTRIBUTION=NORMAL.

Censored Variable Options

CENSORED < (*censored-options***) >**
specifies that the endogenous variables in this statement are censored. Valid *censored-options* are as follows:

LB=*value or variable*
LOWERBOUND=*value or variable*
specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value or variable*
UPPERBOUND=*value or variable*
specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED < (*truncated-options***) >**
specifies that the endogenous variables in this statement are truncated. Valid *truncated-options* are as follows:

LB=*value or variable*
LOWERBOUND=*value or variable*
specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value or variable*
UPPERBOUND=*value or variable*

specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Selection Options

SELECT (*select-option***)**

specifies selection criteria for sample selection model. *Select-option* specifies the condition for the endogenous variable to be selected. It is written as a variable name, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a number:

variable operator number

The *variable* is the endogenous variable that the selection is based on. The *operator* can be =, <, >, <= , or >=. Multiple *select-options* can be combined with the logic operators: AND, OR. This example illustrates the use of SELECT option.

```
endogenous y1 ~ select(z=0);
endogenous y2 ~ select(z=1 and z=2);
```

HETERO Statement

HETERO *dependent variables* ∼ *exogenous variables / options*;

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way these variables are used to model the error variance. The heteroscedastic regression model supported by PROC QLIM is

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$\epsilon_i \sim \mathrm{N}(0, \sigma_\mathrm{i}^2)$$

See the section "Heteroscedasticity" on page 1340 for more details on the specification of functional forms.

LINK=*value*

The functional form can be specified using the LINK= option. The following option values are allowed:

EXP specifies exponential link function

$$\sigma_i^2 \quad = \quad \sigma^2(1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma}))$$

LINEAR specifies linear link function

$$\sigma_i^2 = \sigma^2(1 + \mathbf{z}_i'\gamma)$$

When the LINK= option is not specified, the exponential link function is specified by default.

NOCONST
specifies that there is no constant in linear or exponential heteroscedasticity model.

$$\begin{aligned}\sigma_i^2 &= \sigma^2(\mathbf{z}_i'\gamma) \\ \sigma_i^2 &= \sigma^2\exp(\mathbf{z}_i'\gamma)\end{aligned}$$

SQUARE
estimates the model using the square of linear heteroscedasticity function. For example, you can specify the following heteroscedasticity function:

$$\sigma_i^2 = \sigma^2(1 + (\mathbf{z}_i'\gamma)^2)$$

```
model y = x1 x2 / discrete;
hetero y ~ z1 / link=linear square;
```

The option SQUARE does not apply to exponential heteroscedasticity function because the square of an exponential function of $\mathbf{z}_i'\gamma$ is the same the exponetial of $2\mathbf{z}_i'\gamma$. Hence the only difference is that all γ estimates are divided by two.

INIT Statement

INIT *initvalue1 [, initvalue2 . . .]* **;**

The INIT statement is used to set initial values for parameters in the optimization. Any number of INIT statements can be specified.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equality operator (=), followed by a number:

parameter <=> number

MODEL Statement

MODEL *dependent = regressors / options ;*

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model.

The following options can be used in the MODEL statement after a slash (/).

LIMIT1=*value*
specifies the restriction of the threshold value of the first category when the ordinal probit or logit model is estimated. LIMIT1=ZERO is the default option. When LIMIT1=VARYING is specified, the threshold value is estimated.

NOINT
suppresses the intercept parameter.

Endogenous Variable Options

The endogenous variable options are the same as the options specified in the ENDOGENOUS statement. If an endogenous variable has endogenous option specified in both MODEL statement and ENDOGENOUS statement, the option in ENDOGENOUS statement is used.

BOXCOX Estimation Options

BOXCOX (*option-list***)**
specifies options that are used for Box-Cox regression or regressor transformation. For example, the Box-Cox regression is specified as

```
model y = x1 x2 / boxcox(y=lambda,x1 x2)
```

PROC QLIM estimates the following Box-Cox regression model:

$$y_i^{(\lambda)} = \beta_0 + \beta_1 x_{1i}^{(\lambda_2)} + \beta_2 x_{2i}^{(\lambda_2)} + \epsilon_i$$

The *option-list* takes the form of *variable-list* < = varname > separated by ’,’. The *variable-list* specifies the list of variables to have the same Box-Cox transformation. *varname* specifies the name of this Box-Cox coefficient. If *varname* is not specified, the coefficient is called _Lambda*i* where *i* increments sequentially.

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

PROC QLIM uses the NonLinear Optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options of the NLOPTIONS statement, see Chapter 10, “Nonlinear Optimization Methods.”

OUTPUT Statement

OUTPUT *<OUT=SAS-data-set> <output-options>***;**

The OUTPUT statement creates a new SAS data set containing all variables in the input data set and, optionally, the estimates of $\mathbf{x}'\beta$, predicted value, residual, marginal effects, probability, standard deviation of the error, expected value, conditional expected value, inverse Mills ratio. When the response values are missing for the observation, all output estimates except residual are still computed as long as none of the explanatory variables is missing. This enables you to compute these statistics for prediction. You can only specify one OUTPUT statement.

Details on the specifications in the OUTPUT statement are as follows:

CONDITIONAL
output estimates of conditional expected values of continuous endogenous variables.

ERRSTD
output estimates of σ_j, the standard deviation of the error term.

EXPECTED
output estimates of expected values of continuous endogenous variables.

MARGINAL
output marginal effects.

MILLS
output estimates of inverse Mills ratios of continuous endogenous variables.

OUT=*SAS-data-set*
names the output data set.

PREDICTED
output estimates of predicted endogenous variables.

PROB
output estimates of probability of discrete endogenous variables taking the current observed responses.

PROBALL
output estimates of probability of discrete endogenous variables for all possible responses.

RESIDUAL
output estimates of residuals of continuous endogenous variables.

XBETA
output estimates of $\mathbf{x}'\beta$.

RESTRICT Statement

RESTRICT *restriction1 [, restriction2 ...]* **;**

The RESTRICT statement is used to impose linear restrictions on the parameter estimates. Any number of RESTRICT statements can be specified.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional.

Restriction expressions can be composed of parameter names, times ($*$), plus ($+$) and minus ($-$) operators, and constants. Parameters named in restriction expressions must be among the parameters estimated by the model. The restriction expressions must be a linear function of the parameters.

The following is an example of the use of the RESTRICT statement:

```
proc qlim data=one;
model y = x1-x10 / discrete;
restrict x1*2 <= x2 + x3;
run;
```

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

Details

Ordinal Discrete Choice Modeling

Binary Probit and Logit Model

The binary choice model is

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where the sign of the dependent variable is only observed as follows:

$$\begin{aligned} y_i &= 1 \quad \text{if} \quad \mathrm{y_i^*} > 0 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The disturbance, ϵ_i, of the probit model has standard normal distribution with the distribution function (CDF)

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)dt$$

The disturbance of the logit model has standard logistic distribution with the CDF

$$\Lambda(x) = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$$

The binary discrete choice model has the following probability that the event $\{y_i = 1\}$ occurs:

$$P(y_i = 1) = F(\mathbf{x}_i'\boldsymbol{\beta}) = \begin{cases} \Phi(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(probit)} \\ \Lambda(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(logit)} \end{cases}$$

The log-likelihood function is

$$\ell = \sum_{i=1}^{N} \left\{ y_i \log[F(\mathbf{x}_i'\boldsymbol{\beta})] + (1-y_i)\log[1 - F(\mathbf{x}_i'\boldsymbol{\beta})] \right\}$$

where the CDF $F(x)$ is defined as $\Phi(x)$ for the probit model while $F(x) = \Lambda(x)$ for logit. The first order derivative of the logit model are

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} (y_i - \Lambda(\mathbf{x}_i'\boldsymbol{\beta}))\mathbf{x}_i$$

The probit model has more complicated derivatives

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \left[\frac{(2y_i - 1)\phi((2y_i-1)\mathbf{x}_i'\boldsymbol{\beta})}{\Phi((2y_i-1)\mathbf{x}_i'\boldsymbol{\beta})} \right] \mathbf{x}_i = \sum_{i=1}^{N} r_i \mathbf{x}_i$$

where

$$r_i = \frac{(2y_i - 1)\phi((2y_i-1)\mathbf{x}_i'\boldsymbol{\beta})}{\Phi((2y_i-1)\mathbf{x}_i'\boldsymbol{\beta})}$$

Note that logit maximum likelihood estimates are greater than probit maximum likelihood estimates by approximately $\frac{\pi}{\sqrt{3}}$, since the probit parameter estimates ($\boldsymbol{\beta}$) are standardized and the error term with logistic distribution has a variance of $\frac{\pi^2}{3}$.

Ordinal Probit/Logit

When the dependent variable is observed in sequence with M categories, binary discrete choice modeling is not appropriate for data analysis. McKelvey and Zavoina (1975) proposed the ordinal (or ordered) probit model.

Consider the following regression equation:

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where error disturbances, ϵ_i, have the distribution function F. The unobserved continuous random variable, y_i^*, is identified as M categories. Suppose there are $M+1$ real numbers, $\mu_0, \cdots, \mu_M$, where $\mu_0 = -\infty$, $\mu_1 = 0$, $\mu_M = \infty$, and $\mu_0 \leq \mu_1 \leq \cdots \leq \mu_M$. Define that

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \leq \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The log-likelihood function is

$$\ell = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} \log \left[F(R_{i,j}) - F(R_{i,j-1}) \right]$$

where

$$d_{ij} = \begin{cases} 1 & \text{if } \mu_{j-1} < y_i \leq \mu_j \\ 0 & \text{otherwise} \end{cases}$$

The first derivatives are written as

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} \left[\frac{f(R_{i,j-1}) - f(R_{i,j})}{F(R_{i,j}) - F(R_{i,j-1})} \mathbf{x}_i \right]$$

$$\frac{\partial \ell}{\partial \mu_k} = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} \left[\frac{\delta_{j,k} f(R_{i,j}) - \delta_{j-1,k} f(R_{i,j-1})}{F(R_{i,j}) - F(R_{i,j-1})} \right]$$

where $f(x) = \frac{dF(x)}{dx}$ and $\delta_{j,k} = 1$ if $j = k$. When the ordinal probit is estimated, it is assumed that $F(R_{i,j}) = \Phi(R_{i,j})$. The ordinal logit model is estimated if $F(R_{i,j}) = \Lambda(R_{i,j})$. The first threshold parameter, μ_1, is estimated when the LIMIT1=VARYING option is specified. By default (LIMIT1=ZERO), so that $M-2$ threshold parameters $(\mu_2, \ldots, \mu_{M-1})$ are estimated.

The ordered probit models are analyzed by Aitchison and Silvey (1957), and Cox (1970) discussed ordered response data using the logit model. They defined the probability that y_i^* belongs to jth category as

$$P[\mu_{j-1} < y_i \leq \mu_j] = F(\mu_j + \mathbf{x}_i'\boldsymbol{\theta}) - F(\mu_{j-1} + \mathbf{x}_i'\boldsymbol{\theta})$$

where $\mu_0 = -\infty$ and $\mu_M = \infty$. Therefore, the ordered response model analyzed by Aitchison and Silvey can be estimated if the LIMIT1=VARYING option is specified. Note that $\boldsymbol{\theta} = -\boldsymbol{\beta}$.

Goodness-of-Fit Measures

McFadden (1974) suggested a likelihood ratio index that is analogous to the R^2 in the linear regression model.

$$R_M^2 = 1 - \frac{\ln L}{\ln L_0}$$

where L is the value of the maximum likelihood function and L_0 is a likelihood function when regression coefficients except an intercept term are zero. It can be shown that L_0 can be written as

$$L_0 = \sum_{j=1}^{M} N_j \ln(\frac{N_j}{N}),$$

where N_j is the number of responses in category j.

Estrella (1998) proposes the following requirements for a goodness-of-fit measure to be desirable in discrete choice modeling:

- The measure must take values in $[0, 1]$, where 0 represents no fit and 1 corresponds to perfect fit.
- The measure should be directly related to the valid test statistic for significance of all slope coefficients.
- The derivative of the measure with respect to the test statistic should comply with corresponding derivatives in a linear regression.

Estrella's measure is written

$$R_{E1}^2 = 1 - \left(\frac{\ln L}{\ln L_0}\right)^{-\frac{2}{N}\ln L_0}$$

An alternative measure suggested by Estrella is

$$R_{E2}^2 = 1 - [(\ln L - K)/\ln L_0]^{-\frac{2}{N}\ln L_0}$$

where $\ln L_0$ is computed with null slope parameter values, N is the number observations used, and K represents the number of estimated parameters.

Other goodness-of-fit measures are summarized as follows:

$$R^2_{CU1} = 1 - \left(\frac{L_0}{L}\right)^{\frac{2}{N}} \quad \text{(Cragg-Uhler 1)}$$

$$R^2_{CU2} = \frac{1 - (L_0/L)^{\frac{2}{N}}}{1 - L_0^{\frac{2}{N}}} \quad \text{(Cragg-Uhler 2)}$$

$$R^2_A = \frac{2(\ln L - \ln L_0)}{2(\ln L - \ln L_0) + N} \quad \text{(Aldrich-Nelson)}$$

$$R^2_{VZ} = R^2_A \frac{2\ln L_0 - N}{2\ln L_0} \quad \text{(Veall-Zimmermann)}$$

$$R^2_{MZ} = \frac{\sum_{i=1}^N (\hat{y}_i - \bar{\hat{y}}_i)^2}{N + \sum_{i=1}^N (\hat{y}_i - \bar{\hat{y}}_i)^2} \quad \text{(McKelvey-Zavoina)}$$

where $\hat{y}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ and $\bar{\hat{y}}_i = \sum_{i=1}^N \hat{y}_i/N$.

Limited Dependent Variable Models

Censored Regression Models

When the dependent variable is censored, values in a certain range are all transformed to a single value. For example, the standard tobit model can be defined as

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \le 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$. The log-likelihood function of the standard censored regression model is

$$\ell = \sum_{i\in\{y_i=0\}} \ln[1 - \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma)] + \sum_{i\in\{y_i>0\}} \ln\left[\frac{\phi((y_i - \mathbf{x}_i'\boldsymbol{\beta})/\sigma)}{\sigma}\right]$$

where $\Phi(\cdot)$ is the cumulative density function of the standard normal distribution and $\phi(\cdot)$ is the probability density function of the standard normal distribution.

The tobit model can be generalized to handle observation-by-observation censoring. The censored model on both of the lower and upper limits can be defined as follows

$$y_i = \begin{cases} R_i & \text{if } y_i^* \geq R_i \\ y_i^* & \text{if } L_i < y_i^* < R_i \\ L_i & \text{if } y_i^* \leq L_i \end{cases}$$

The log-likelihood function can be written as

$$\begin{aligned} \ell &= \sum_{i \in \{L_i < y_i < R_i\}} \ln \phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma + \sum_{i \in \{y_i = R_i\}} \ln \Phi(-\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) + \\ & \sum_{i \in \{y_i = L_i\}} \ln \Phi(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) \end{aligned}$$

Log-likelihood functions of the lower- or upper-limit censored model are easily derived from the two-limit censored model. The log-likelihood function of the lower-limit censored model is

$$\ell = \sum_{i \in \{y_i > L_i\}} \ln \phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma + \sum_{i \in \{y_i = L_i\}} \ln \Phi(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})$$

The log-likelihood function of the upper-limit censored model is

$$\ell = \sum_{i \in \{y_i < R_i\}} \ln \phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma + \sum_{i \in \{y_i = R_i\}} \ln \left[1 - \Phi(\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})\right]$$

Truncated Regression Models

In truncated model, the observed sample is a subset of the population where the dependent variable falls in a certain range. For example, when neither a dependent variable nor exogenous variables are observed for $y_i^* \leq 0$, the truncated regression model can be specified. The log-likelihood function of the truncated regression model is

$$\ell = \sum_{i \in \{y_i > 0\}} \left\{ -\ln \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma) + \ln \left[\frac{\phi((y_i - \mathbf{x}_i'\boldsymbol{\beta})/\sigma)}{\sigma}\right] \right\}$$

The two-limit truncation model is defined as

$$y_i = y_i^* \quad \text{if } L_i < y_i^* < R_i$$

The log-likelihood function of the two-limit truncated regression model is

$$\ell = \sum_{i=1}^{N} \left\{ \ln \phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma - \ln \left[\Phi(\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) - \Phi(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})\right] \right\}$$

The log-likelihood functions of the lower- and upper-limit truncation model are

$$\begin{aligned}\ell &= \sum_{i=1}^{N}\left\{\ln\left[\phi\left(\frac{y_i-\mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right]-\ln\left[1-\Phi\left(\frac{L_i-\mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right]\right\} \quad \text{(lower)}\\ \ell &= \sum_{i=1}^{N}\left\{\ln\left[\phi\left(\frac{y_i-\mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right]-\ln\left[\Phi\left(\frac{R_i-\mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right]\right\} \quad \text{(upper)}\end{aligned}$$

Heteroscedasticity and Box-Cox Transformation

Heteroscedasticity

If the variance of regression disturbance (ϵ_i) is heteroscedastic, the variance can be specified as a function of variables

$$E(\epsilon_i^2) = \sigma_i^2 = f(\mathbf{z}_i'\boldsymbol{\gamma})$$

The following table shows various functional forms of heteroscedasticity and the corresponding options to request each model.

No.	Model	Options
1	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\exp(\mathbf{z}_i'\gamma))$	link=EXP (default)
2	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2\exp(\mathbf{z}_i'\gamma)$	link=EXP noconst
3	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\sum_{l=1}^{L}\gamma_l z_{li})$	link=LINEAR
4	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+(\sum_{l=1}^{L}\gamma_l z_{li})^2)$	link=LINEAR square
5	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(\sum_{l=1}^{L}\gamma_l z_{li})$	link=LINEAR noconst
6	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2((\sum_{l=1}^{L}\gamma_l z_{li})^2)$	link=LINEAR square noconst

For discrete choice models, σ^2 is normalized ($\sigma^2 = 1$) since this parameter is not identified. Note that in models 3 and 5, it may be possible that variances of some observations are negative. Although the QLIM procedure assigns a large penalty to move the optimization away from such regions, sometimes the optimization may stuck in such regions. Signs of such outcome include extremely small likelihood values or missing standard errors in the estimates. In models 2 and 6, variances are guaranteed to be greater or equal to zero but it may be possible that variances of some observations are very close to zero. In these scenarios, standard errors may be missing. Models 1 and 4 do not have such problems. Variances in these models are always positive and never close to zero.

The heteroscedastic regression model is estimated using the following log-likelihood function:

$$\ell = -\frac{N}{2}\ln(2\pi) - \sum_{i=1}^{N}\frac{1}{2}\ln(\sigma_i^2) - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{e_i}{\sigma_i}\right)^2$$

where $e_i = y_i - \mathbf{x}_i'\boldsymbol{\beta}$.

Box-Cox Modeling

The Box-Cox transformation on x is defined as

$$x^{(\lambda)} = \begin{cases} \frac{x^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(x) & \text{if } \lambda = 0 \end{cases}$$

The Box-Cox regression model with heteroscedasticity is written

$$\begin{aligned} y_i^{(\lambda_0)} &= \beta_0 + \sum_{k=1}^{K} \beta_k x_{ki}^{(\lambda_k)} + \epsilon_i \\ &= \mu_i + \epsilon_i \end{aligned}$$

where $\epsilon_i \sim N(0, \sigma_i^2)$ and transformed variables must be positive. In practice, too many transformation parameters cause numerical problems in model fitting. It is common to have the same Box-Cox transformation performed on all the variables, that is, $\lambda_0 = \lambda_1 = \cdots = \lambda_K$. It is required for the magnitude of transformed variables to be in the tolerable range if the corresponding transformation parameters are $|\lambda| > 1$.

The log-likelihood function of the Box-Cox regression model is written

$$\ell = -\frac{N}{2}\ln(2\pi) - \sum_{i=1}^{N} \ln(\sigma_i) - \frac{1}{2\sigma_i^2}\sum_{i=1}^{N} e_i^2 + (\lambda_0 - 1)\sum_{i=1}^{N} \ln(y_i)$$

where $e_i = y_i^{(\lambda_0)} - \mu_i$.

When the dependent variable is discrete, censored, or truncated, the Box-Cox transformation can only be applied to explanatory variables.

Bivariate Limited Dependent Variable Modeling

The generic form of a bivariate limited dependent variable model is

$$\begin{aligned} y_{1i}^* &= \mathbf{x}_{1i}'\boldsymbol{\beta}_1 + \epsilon_{1i} \\ y_{2i}^* &= \mathbf{x}_{2i}'\boldsymbol{\beta}_2 + \epsilon_{2i} \end{aligned}$$

where the disturbances, ϵ_{1i} and ϵ_{2i}, have normal distribution with zero mean, standard deviations σ_1 and σ_2, and correlation of ρ. y_1^* and y_2^* are latent variables. The dependent variables y_1 and y_2 are observed if the latent variables y_1^* and y_2^* fall in certain ranges.

$$\begin{aligned} y_1 = y_{1i} \quad & \text{if} \quad y_{1i}^* \in D_1(y_{1i}) \\ y_2 = y_{2i} \quad & \text{if} \quad y_{2i}^* \in D_2(y_{2i}) \end{aligned}$$

D is a transformation from (y_{1i}^*, y_{2i}^*) to (y_{1i}, y_{2i}). For example, if $y1$ and $y2$ are censored variables with lower bound 0, then

$$
\begin{array}{llll}
y_1 = y_{1i} & \text{if } \; y_{1i}^* > 0, & y_1 = 0 & \text{if } \; y_{1i}^* \leq 0 \\
y_2 = y_{2i} & \text{if } \; y_{2i}^* > 0, & y_2 = 0 & \text{if } \; y_{2i}^* \leq 0
\end{array}
$$

There are three cases for the log likelihood of (y_{1i}, y_{2i}). The first case is that $y_{1i} = y_{1i}^*$ and $y_{2i} = y_{2i}^*$. That is, this observation is mapped to one point in the space of latent variables. The log likelihood is computed from a bivariate normal density,

$$
\ell_i = \ln \phi_2\left(\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}, \frac{y_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}, \rho\right) - \ln \sigma_1 - \ln \sigma_2
$$

where $\phi_2(u, v, \rho)$ is the density function for standardized bivariate normal distribution with correlation ρ,

$$
\phi_2(u, v, \rho) = \frac{e^{-(1/2)(u^2 + v^2 - 2\rho uv)/(1-\rho^2)}}{2\pi(1-\rho^2)^{1/2}}
$$

The second case is that one observed dependent variable is mapped to a point of its latent variable and the other dependent variable is mapped to a segment in the space of its latent variable. For example, in the bivariate censored model specified, if observed $y1 > 0$ and $y2 = 0$, then $y1^* = y1$ and $y2^* \in (-\infty, 0]$. In general, the log likelihood for one observation can be written as follows (the subscript i is dropped for simplicity): If one set is a single point and the other set is a range, without loss of generality, let $D_1(y_1) = \{y_1\}$ and $D_2(y_2) = [L_2, R_2]$,

$$
\begin{aligned}
\ell_i &= \ln \phi\left(\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}\right) - \ln \sigma_1 \\
&+ \ln\left[\Phi\left(\frac{R_2 - \mathbf{x_2}'\boldsymbol{\beta}_2 - \rho\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}{\sigma_2}\right) - \Phi\left(\frac{L_2 - \mathbf{x_2}'\boldsymbol{\beta}_2 - \rho\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}{\sigma_2}\right)\right]
\end{aligned}
$$

where ϕ and Φ are the density function and the cumulative probability function for standardized univariate normal distribution.

The third case is that both dependent variables are mapped to segments in the space of latent variables. For example, in the bivariate censored model specified, if observed $y1 = 0$ and $y2 = 0$, then $y1^* \in (-\infty, 0]$ and $y2^* \in (-\infty, 0]$. In general, if $D_1(y_1) = [L_1, R_1]$ and $D_2(y_2) = [L_2, R_2]$, the log likelihood is

$$
\ell_i = \ln \int_{\frac{L_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}^{\frac{R_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}} \int_{\frac{L_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}}^{\frac{R_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}} \phi_2(u, v, \rho)\, du\, dv
$$

Selection Models

In sample selection models, one or several dependent variables are observed when another variable takes certain values. For example, the standard Heckman selection model can be defined as

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i \text{ if } z_i = 1$$

where u_i and ϵ_i are jointly normal with zero mean, standard deviations of 1 and σ, and correlation of ρ. z is the variable that the selection is based on and y is observed when z has a value of 1. Least squares regression using the observed data of y produces inconsistent estimates of $\boldsymbol{\beta}$. Maximum likelihood method is used to estimate selection models. The log-likelihood function of the Heckman selection model is written as

$$\begin{aligned} \ell &= \sum_{i\in\{z_i=0\}} \ln[1-\Phi(\mathbf{w}_i'\boldsymbol{\gamma})] \\ &+ \sum_{i\in\{z_i=1\}} \{\ln\phi(\frac{y_i-\mathbf{x_i}'\boldsymbol{\beta}}{\sigma}) - \ln\sigma + \ln\Phi(\mathbf{w}_i'\boldsymbol{\gamma} + \rho\frac{y_i-\mathbf{x_i}'\boldsymbol{\beta}}{\sigma})\} \end{aligned}$$

Only one variable is allowed for the selection to be based on, but the selection may lead to several variables. For example, in the following switch regression model,

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$\begin{aligned} y_{1i} &= \mathbf{x}_{1i}'\boldsymbol{\beta}_1 + \epsilon_{1i} \text{ if } z_i = 0 \\ y_{2i} &= \mathbf{x}_{2i}'\boldsymbol{\beta}_2 + \epsilon_{2i} \text{ if } z_i = 1 \end{aligned}$$

z is the variable that the selection is based on. If $z = 0$, then y_1 is observed. If $z = 1$, then y_2 is observed. Because it is never the case that $y1$ and $y2$ are observed at the same time, the correlation between y_1 and y_2 cannot be estimated. Only the correlation between z and y_1 and the correlation between z and y_2 can be estimated. This estimation uses the maximum likelihood method.

Multivariate Limited Dependent Models

The multivariate model is similar to bivariate models. The generic form of the multivariate limited dependent variable model is

$$\begin{aligned} y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + \epsilon_{1i} \\ y^*_{2i} &= \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + \epsilon_{2i} \\ &\ldots \\ y^*_{mi} &= \mathbf{x}'_{mi}\boldsymbol{\beta}_m + \epsilon_{mi} \end{aligned}$$

where m is the number of models to be estimated. The vector ϵ has multivariate normal distribution with mean 0 and variance-covariance matrix Σ. Similar to bivariate models, the likelihood may involve computing multivariate normal integrations. This is done using Monte Carlo integration. (See Genz 1992; Hajivassiliou and McFadden D. 1998).

When the number of equations N increases in a system, the number of parameters increases at the rate of N^2 because of the correlation matrix. When the number of parameters is large, sometimes the optimization converges but some of the standard deviations are missing. This usually means that the model is overparameterized. The default method for computing the covariance is to use the inverse Hessian matrix. Hessian is computed by finite differences, and in overparameterized cases, the inverse cannot be computed. It is recommended to reduce the number of parameters in such cases. Sometimes using the outer product covariance matrix (COVEST=OP option) may also help.

Output

XBeta, Predicted, Residual

Xbeta is the structural part on the right-hand side of the model. Predicted value is the predicted dependent variable value. For censored variables, if the predicted value is outside the boundaries, it is reported as the closest boundary. For discrete variables, it is the level whose boundaries Xbeta falls in between. Residual is only defined for continuous variables and is defined as

$$Residual = Observed - Predicted$$

Error Standard Deviation

Error standard deviation is σ_i in the model. It only varies when the HETERO statement is used.

Marginal Effects

Marginal effect is the contribution of one control variable to the response variable. For the binary choice model and ordinal response model with M categories, specify

$M+1$ real numbers, $\mu_0, \cdots, \mu_M$, where $\mu_0 = -\infty$, $\mu_1 = 0$ (or estimated when LIMIT1=VARYING), $\mu_M = \infty$, and $\mu_0 \leq \mu_1 \leq \cdots \leq \mu_M$. Define that

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written

$$P[\mu_{j-1} < y_i^* \leq \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The marginal effect of changes in the regressors on the probability of $y_i = j$ is then

$$\frac{\partial Prob[y_i = j]}{\partial \mathbf{x}} = [f(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta}) - f(\mu_j - \mathbf{x}_i'\boldsymbol{\beta})]\boldsymbol{\beta}$$

where $f(x) = \frac{dF(x)}{dx}$. In particular,

$$f(x) = \frac{dF(x)}{dx} = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{(probit)} \\ \frac{e^{-x}}{[1+e^{(-x)}]^2} & \text{(logit)} \end{cases}$$

The marginal effects in the Box-Cox regression model are

$$\frac{\partial E[y_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \frac{x^{\lambda_k - 1}}{y^{\lambda_0 - 1}}$$

The marginal effects in the truncated regression model are

$$\frac{\partial E[y_i | L_i < y_i^* < R_i]}{\partial \mathbf{x}} = \boldsymbol{\beta}\left[1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i\phi(a_i) - b_i\phi(b_i)}{\Phi(b_i) - \Phi(a_i)}\right]$$

where $a_i = \frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$.

The marginal effects in the censored regression model are

$$\frac{\partial E[y|\mathbf{x}_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \times Prob[L_i < y_i^* < R_i]$$

Inverse Mills Ratio, Expected and Conditionally Expected Values

These values only apply to continuous variables. Let L_i and R_i be the lower boundary and upper boundary for the y_i. Define $a_i = \frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$. Then the Inverse Mills Ratio is defined as

$$\lambda = \frac{(\phi(a_i) - \phi(b_i))}{(\Phi(b_i) - \Phi(a_i))}$$

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(\Phi(a_i) - \Phi(b_i) + (1 - \Phi(b_i))R_i$$

For a noncensored variable, this is

$$E[y_i] = \mathbf{x}_i'\boldsymbol{\beta}$$

The conditional expected value is the expectation given that the variable is inside the boundaries.

$$E[y_i|L_i < y_i < R_i] = \mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i$$

Probability

Probability is only for discrete responses. It is the marginal probability that the discrete response is taking the value of the observation. If the PROBALL option is specified, then the probability for all of the possible responses of the discrete variables are computed.

Naming

Naming of Parameters

When there is only one equation in the estimation, parameters are named in the same way as other SAS procedures such as REG, PROBIT, etc. The constant in the regression equation is called Intercept. The coefficients on independent variables are named by the independent variables. The standard deviation of the errors is called _Sigma. If there are Box-Cox transformations, the coefficients are named _Lambda$i$, where i increments from 1, or as specified by the user. The limits for the discrete dependent variable are named _Limit$i$. If the LIMIT=varying option is specified, then _Limit$i$ starts from 1. If the LIMIT=varying option is not specified, then _Limit1 is set to 0 and the limit parameters start from $i = 2$. If the HETERO statement is included, the coefficients of the independent variables in the hetero equation are called _H.x where x is the name of the independent variable.

When there are multiple equations in the estimation, the parameters in the main equation are named in the format of $y.x$ where y is the name of the dependent variable and x is the name of the independent variable. The standard deviation of the errors is call _Sigma.y. Box-Cox parameters are called _Lambda$i.y$ and limit variables are called _Limit$i.y$. Parameters in the HETERO statement are named as _H.$y.x$. In the OUTEST= data set, all variables are changed from '.' to '_'.

Naming of Output Variables

The following table shows the option in the Output statement, with the corresponding variable names and their explanation.

Option	Name	Explanation
PREDICTED	P_y	Predicted value of y
RESIDUAL	RESID_y	Residual of y, (y-PredictedY)
XBETA	XBETA_y	Structure part ($\mathbf{x}'\boldsymbol{\beta}$) of y equation
ERRSTD	ERRSTD_y	Standard deviation of error term
PROB	PROB_y	Probability that y is taking the observed value in this observation. (Discrete y only)
PROBALL	PROBi_y	Probability that y is taking the i^{th} value. (Discrete y only)
MILLS	MILLS_y	Inverse Mills ratio for y
EXPECTED	EXPCT_y	Unconditional expected value of y
CONDITIONAL	CEXPCT_y	Conditional expected value of y, condition on the truncation.
MARGINAL	MEFF_x	Marginal effect of x on y ($\frac{\partial y}{\partial x}$) with single equation
	MEFF_y_x	Marginal effect of x on y ($\frac{\partial y}{\partial x}$) with multiple equations
	MEFF_Pi_x	Marginal effect of x on y ($\frac{\partial Prob(y=i)}{\partial x}$) with single equation and discrete y
	MEFF_Pi_y_x	Marginal effect of x on y ($\frac{\partial Prob(y=i)}{\partial x}$) with multiple equations and discrete y

If you prefer to name the output variables differently, you can use the RENAME option in the data set. For example, the following statement renames the residual of y to *Resid*.

```
proc qlim data=one;
model y = x1-x10 / censored;
output out=outds(rename=(resid_y=resid)) residual;
run;
```

ODS Table Names

PROC QLIM assigns a name to each table it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 22.1. ODS Tables Produced in PROC QLIM

ODS Table Name	Description	Option
	ODS Tables Created by the Model Statement	
FitSummary	Summary of Nonlinear Estimation	default
ResponseProfile	Response Profile	default
GoodnessOfFit	Pseudo-R^2 Measures	default

Table 22.1. (continued)

ODS Table Name	Description	Option
ParameterEstimates	Parameter Estimates	default
CovB	Covariance of Parameter Estimates	COVB
CorrB	Correlation of Parameter Estimates	CORRB

Examples

Example 22.1. Ordered Data Modeling

Cameron and Trivedi (1986) studied an Australian Health Survey data. Variable definitions are given in Cameron and Trivedi (1998, p. 68). The dependent variable, dvisits, has nine ordered values. The following SAS statements estimate the ordinal probit model:

```
data docvisit;
   input sex age agesq income levyplus freepoor freerepa
         illness actdays hscore chcond1 chcond2 dvisits;
   y = (dvisits > 0);
   if ( dvisits > 8 ) then dvisits = 8;

  ...

 0 0.72 0.5184 0.25  0  0  1  0  0  0  0  0  0
;



title1 "Ordered Discrete Responses";
proc qlim data=docvisit;
  model dvisits = sex age agesq income levyplus
   freepoor freerepa illness actdays hscore
   chcond1 chcond2 / discrete;
run;
```

The output of the QLIM procedure for Ordered Data Modeling is shown in Output 22.1.1.

Output 22.1.1. Ordered Data Modeling

```
                    Ordered Discrete Responses

                       The QLIM Procedure

               Discrete Response Profile of dvisits

        Index         Value           Frequency    Percent

          1             0                4141       79.79
          2             1                 782       15.07
          3             2                 174        3.35
          4             3                  30        0.58
          5             4                  24        0.46
          6             5                   9        0.17
          7             6                  12        0.23
          8             7                  12        0.23
          9             8                   6        0.12
```

Output 22.1.1. (continued)

```
                           Ordered Discrete Responses

                               Model Fit Summary

                  Number of Endogenous Variables            1
                  Endogenous Variable                  dvisits
                  Number of Observations                  5190
                  Log Likelihood                         -3138
                  Maximum Absolute Gradient          0.0006844
                  Number of Iterations                      88
                  AIC                                     6316
                  Schwarz Criterion                       6447


                            Goodness-of-Fit Measures

   Measure                      Value    Formula

   Likelihood Ratio (R)        789.73    2 * (LogL - LogL0)
   Upper Bound of R (U)        7065.9    - 2 * LogL0
   Aldrich-Nelson              0.1321    R / (R+N)
   Cragg-Uhler 1               0.1412    1 - exp(-R/N)
   Cragg-Uhler 2               0.1898    (1-exp(-R/N)) / (1-exp(-U/N))
   Estrella                     0.149    1 - (1-R/U)^(U/N)
   Adjusted Estrella           0.1416    1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
   McFadden's LRI              0.1118    R / U
   Veall-Zimmermann            0.2291    (R * (U+N)) / (U * (R+N))
   McKelvey-Zavoina            0.2036

   N = # of observations, K = # of regressors


                               Parameter Estimates

                                       Standard                 Approx
        Parameter        Estimate         Error    t Value    Pr > |t|

        Intercept       -1.378704      0.147412      -9.35      <.0001
        sex              0.131885      0.043785       3.01      0.0026
        age             -0.534198      0.815897      -0.65      0.5126
        agesq            0.857317      0.898353       0.95      0.3399
        income          -0.062211      0.068017      -0.91      0.3604
        levyplus         0.137031      0.053262       2.57      0.0101
        freepoor        -0.346045      0.129638      -2.67      0.0076
        freerepa         0.178382      0.074348       2.40      0.0164
        illness          0.150485      0.015747       9.56      <.0001
        actdays          0.100575      0.005850      17.19      <.0001
        hscore           0.031862      0.009201       3.46      0.0005
        chcond1          0.061602      0.049024       1.26      0.2089
        chcond2          0.135322      0.067711       2.00      0.0457
        _Limit2          0.938884      0.031219      30.07      <.0001
        _Limit3          1.514288      0.049329      30.70      <.0001
        _Limit4          1.711660      0.058148      29.44      <.0001
        _Limit5          1.952860      0.072010      27.12      <.0001
        _Limit6          2.087422      0.081643      25.57      <.0001
        _Limit7          2.333787      0.101746      22.94      <.0001
        _Limit8          2.789795      0.156177      17.86      <.0001
```

By default, ordinal probit/logit models are estimated assuming that the first threshold or limit parameter (μ_1) is 0. However, this parameter can also be estimated when the

LIMIT1=VARYING option is specified. The probability that y_i^* belongs to the jth category is defined as

$$P[\mu_{j-1} < y_i^* < \mu_j] = F(\mu_j - \mathbf{x}_i'\boldsymbol{\beta}) - F(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta})$$

where $F(\cdot)$ is the logistic or standard normal CDF, $\mu_0 = -\infty$ and $\mu_9 = \infty$. Output 22.1.2 lists ordinal or cumulative logit estimates. Note that the intercept term is suppressed for model identification when μ_1 is estimated.

Output 22.1.2. Ordinal Probit Parameter Estimates with LIMIT1=VARYING

```
                         Ordered Discrete Responses

                             The QLIM Procedure

                            Parameter Estimates

                                       Standard                      Approx
   Parameter          Estimate            Error    t Value    Pr > |t|

   sex                0.131885         0.043785       3.01      0.0026
   age               -0.534187         0.815944      -0.65      0.5127
   agesq              0.857306         0.898403       0.95      0.3400
   income            -0.062211         0.068018      -0.91      0.3604
   levyplus           0.137031         0.053262       2.57      0.0101
   freepoor          -0.346045         0.129638      -2.67      0.0076
   freerepa           0.178382         0.074348       2.40      0.0164
   illness            0.150485         0.015747       9.56      <.0001
   actdays            0.100575         0.005850      17.19      <.0001
   hscore             0.031862         0.009201       3.46      0.0005
   chcond1            0.061602         0.049024       1.26      0.2089
   chcond2            0.135321         0.067711       2.00      0.0457
   _Limit1            1.378705         0.147419       9.35      <.0001
   _Limit2            2.317589         0.150209      15.43      <.0001
   _Limit3            2.892994         0.155200      18.64      <.0001
   _Limit4            3.090366         0.158249      19.53      <.0001
   _Limit5            3.331566         0.164040      20.31      <.0001
   _Limit6            3.466128         0.168746      20.54      <.0001
   _Limit7            3.712493         0.179694      20.66      <.0001
   _Limit8            4.168501         0.215683      19.33      <.0001
```

Example 22.2. Tobit Analysis

The following table shows a subset of the Mroz (1987) data set. In this data, Hours is the number of hours the wife worked outside the household in a given year, Yrs_Ed is the years of education, and Yrs_Exp is the years of work experience. A Tobit model will be fit to the hours worked with years of education and experience as covariates.

```
title1 "Estimating a tobit model";

data subset;
      input Hours Yrs_Ed Yrs_Exp @@;
      if Hours eq 0
         then Lower=.;
         else Lower=Hours;
```

```
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;
run;


proc qlim data=subset;
  model hours = yrs_ed yrs_exp;
  endogenous hours ~ censored(lb=0);
run;
```

The output of the QLIM procedure is shown in Output 22.2.1.

Output 22.2.1. Tobit Analysis

```
                          Estimating a tobit model

                             The QLIM Procedure

                             Model Fit Summary

                Number of Endogenous Variables             1
                Endogenous Variable                    hours
                Number of Observations                    17
                Log Likelihood                     -74.93700
                Maximum Absolute Gradient         1.18953E-6
                Number of Iterations                      23
                AIC                                157.87400
                Schwarz Criterion                  161.20685


                            Parameter Estimates

                                        Standard                 Approx
      Parameter        Estimate            Error    t Value    Pr > |t|

      Intercept    -5598.295129        27.692304    -202.16      <.0001
      Yrs_Ed         373.123254        53.989108       6.91      <.0001
      Yrs_Exp         63.336247        36.551332       1.73      0.0831
      _Sigma        1582.859635       390.074877       4.06      <.0001
```

Example 22.3. Bivariate Probit Analysis

This example shows how to estimate a bivariate probit model. Note the INIT statement in the code that set the initial values for some parameters in the optimization.

```
data a;
  keep y1 y2 x1 x2;
  do i = 1 to 500;
    x1 = rannor( 19283 );
    x2 = rannor( 98721 );
```

```
         u1 = rannor( 76527 );
         u2 = rannor( 65721 );
         y1l = 1 + 2 * x1 + 3 * x2 + u1;
         y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
         if ( y1l > 0 ) then y1 = 1;
         else  y1 = 0;
         if ( y2l > 0 ) then y2 = 1;
         else  y2 = 0;
         output;
        end;
      run;


   proc qlim data=a method=qn;
     init y1.x1 2.8, y1.x2 2.1,
        _rho .1;
     model y1 = x1 x2;
     model y2 = x1 x2;
     endogenous y1 y2 ~ discrete;
   run;
```

The output of the QLIM procedure is shown in Output 22.3.1.

Output 22.3.1. Bivariate Probit Analysis

```
                               The QLIM Procedure

                               Model Fit Summary

                  Number of Endogenous Variables            2
                  Endogenous Variable                   y1 y2
                  Number of Observations                  500
                  Log Likelihood                   -134.90796
                  Maximum Absolute Gradient        3.23486E-7
                  Number of Iterations                     17
                  AIC                               283.81592
                  Schwarz Criterion                 313.31817


                              Parameter Estimates

                                        Standard                 Approx
      Parameter           Estimate         Error    t Value    Pr > |t|

      y1.Intercept        1.003639      0.153677       6.53      <.0001
      y1.x1               2.244374      0.256058       8.77      <.0001
      y1.x2               3.273441      0.341576       9.58      <.0001
      y2.Intercept        3.621164      0.457164       7.92      <.0001
      y2.x1               4.551525      0.576533       7.89      <.0001
      y2.x2              -2.442769      0.332290      -7.35      <.0001
      _Rho                0.144097      0.336458       0.43      0.6685
```

Example 22.4. Sample Selection Model

The following example illustrates the use of PROC QLIM for sample selection models. The data set is the same one from Mroz (1987). The goal is to estimate a wage offer function for married women, accounting for potential selection bias. Of the 753 women, the wage is observed for 428 working women. The labor force participation equation estimated in the introductory example is used for selection. The wage equation use log wage (lwage) as dependent variable. The explanatory variables in the wage equation are the woman's years of schooling (educ), wife's labor experience (exper), square of experience (expersq). The program is illustrated below.

```
proc qlim data=mroz;
  model inlf = nwifeinc educ exper expersq age kidslt6 kidsge6 /discrete;
  model lwage = educ exper expersq / select(inlf=1);
run;
```

The output of the QLIM procedure is shown in Output 22.4.1.

Output 22.4.1. Sample Selection

```
                                The QLIM Procedure

                                Model Fit Summary

                   Number of Endogenous Variables              2
                   Endogenous Variable                inlf lwage
                   Number of Observations                    753
                   Log Likelihood                     -832.88509
                   Maximum Absolute Gradient             0.00524
                   Number of Iterations                       81
                   AIC                                      1694
                   Schwarz Criterion                        1759


                               Parameter Estimates

                                            Standard                 Approx
     Parameter               Estimate          Error    t Value    Pr > |t|

     lwage.Intercept        -0.552698       0.260378      -2.12      0.0338
     lwage.educ              0.108350       0.014861       7.29      <.0001
     lwage.exper             0.042837       0.014879       2.88      0.0040
     lwage.expersq          -0.000837       0.000417      -2.01      0.0449
     _Sigma.lwage            0.663397       0.022707      29.21      <.0001
     inlf.Intercept          0.266450       0.508958       0.52      0.6006
     inlf.nwifeinc          -0.012132       0.004877      -2.49      0.0129
     inlf.educ               0.131341       0.025382       5.17      <.0001
     inlf.exper              0.123282       0.018724       6.58      <.0001
     inlf.expersq           -0.001886       0.000600      -3.14      0.0017
     inlf.age               -0.052829       0.008479      -6.23      <.0001
     inlf.kidslt6           -0.867397       0.118651      -7.31      <.0001
     inlf.kidsge6            0.035873       0.043475       0.83      0.4093
     _Rho                    0.026605       0.147078       0.18      0.8565
```

Note the correlation estimate is insignificant. This indicates that selection bias is not a big problem in the estimation of wage equation.

References

Abramowitz, M. and Stegun, A. (1970), *Handbook of Mathematical Functions*, New York: Dover Press.

Aitchison, J. and Silvey, S. (1957), “The Generalization of Probit Analysis to the Case of Multiple Responses,” *Biometrika*, 44, 131–140.

Amemiya, T. (1978), “The Estimation of a Simultaneous Equation Generalized Probit Model,” *Econometrica*, 46, 1193–1205.

Amemiya, T. (1978), “On a Two-Step Estimate of a Multivariate Logit Model,” *Journal of Econometrics*, 8, 13–21.

Amemiya, T. (1981), “Qualitative Response Models: A Survey,” *Journal of Economic Literature*, 19, 483–536.

Amemiya, T. (1984), “Tobit Models: A Survey,” *Journal of Econometrics*, 24, 3–61.

Amemiya, T. (1985), *Advanced Econometrics*, Cambridge: Harvard University Press.

Ben-Akiva, M. and Lerman, S.R. (1987), *Discrete Choice Analysis*, Cambridge: MIT Press.

Bera, A.K., Jarque, C.M., and Lee, L.-F. (1984), “Testing the Normality Assumption in Limited Dependent Variable Models,” *International Economic Review*, 25, 563–578.

Bloom, D.E. and Killingsworth, M.R. (1985), “Correcting for Truncation Bias Caused by a Latent Truncation Variable,” *Journal of Econometrics*, 27, 131–135.

Box, G.E.P. and Cox, D.R. (1964), “An Analysis of Transformations,” *Journal of the Royal Statistical Society, Series B.*, 26, 211–252.

Cameron, A.C. and Trivedi, P.K. (1986), “Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators,” *Journal of Applied Econometrics*, 1, 29–53.

Cameron, A.C. and Trivedi, P.K. (1998), *Regression Analysis of Count Data*, Cambridge: Cambridge University Press.

Copley, P.A., Doucet, M.S., and Gaver, K.M. (1994), “A Simultaneous Equations Analysis of Quality Control Review Outcomes and Engagement Fees for Audits of Recipients of Federal Financial Assistance,” *The Accounting Review*, 69, 244–256.

Cox, D.R. (1970), *Analysis of Binary Data*, London: Metheun.

Cox, D.R. (1972), “Regression Models and Life Tables,” *Journal of the Royal Statistical Society, Series B*, 20, 187–220.

Cox, D.R. (1975), “Partial Likelihood,” *Biometrika*, 62, 269–276.

Deis, D.R. and Hill, R.C. (1998), “An Application of the Bootstrap Method to the Simultaneous Equations Model of the Demand and Supply of Audit Services,” *Contemporary Accounting Research*, 15, 83–99.

Estrella, A. (1998), "A New Measure of Fit for Equations with Dichotomous Dependent Variables," *Journal of Business and Economic Statistics*, 16, 198–205.

Genz, A. (1992), "Numerical Computation of Multivariate Normal Probabilities," *Journal of Computational and Graphical Statistics*, 1, 141–150.

Godfrey, L.G. (1988), *Misspecification Tests in Econometrics*, Cambridge: Cambridge University Press.

Gourieroux, C., Monfort, A., Renault, E., and Trognon, A. (1987), "Generalized Residuals," *Journal of Econometrics*, 34, 5–32.

Green, W.H. (1997), *Econometric Analysis*, Upper Saddle River, N.J.: Prentice Hall.

Hajivassiliou, V.A. (1993), "Simulation Estimation Methods for Limited Dependent Variable Models," in *Handbook of Statistics*, Vol. 11, ed. G.S. Maddala, C.R. Rao, and H.D. Vinod, New York: Elsevier Science Publishing.

Hajivassiliou, V.A., and McFadden, D. (1998), "The Method of Simulated Scores for the Estimation of LDV Models," *Econometrica*, 66, 863–896.

Heckman, J.J. (1978), "Dummy Endogenous Variables in a Simultaneous Equation System," *Econometrica*, 46, 931–959.

Hinkley, D.V. (1975), "On Power Transformations to Symmetry," *Biometrika*, 62, 101–111.

Kim, M. and Hill, R.C. (1993), "The Box-Cox Transformation-of-Variables in Regression," *Empirical Economics*, 18, 307–319.

King, G. (1989b), *Unifying Political Methodology: The Likelihood Theory and Statistical Inference*, Cambridge: Cambridge University Press.

Lee, L.-F. (1981), "Simultaneous Equations Models with Discrete and Censored Dependent Variables," in *Structural Analysis of Discrete Data with Econometric Applications*, ed. C.F. Manski and D. McFadden, Cambridge: MIT Press

Long, J.S. (1997), *Regression Models for Categorical and Limited Dependent Variables*, Thousand Oaks, CA: Sage Publications, Inc.

McFadden, D. (1974), "Conditional Logit Analysis of Qualitative Choice Behavior," in *Frontiers in Econometrics*, ed. P. Zarembka, New York: Academic Press.

McFadden, D. (1981), "Econometric Models of Probabilistic Choice," in *Structural Analysis of Discrete Data with Econometric Applications*, ed. C.F. Manski and D. McFadden, Cambridge: MIT Press.

McKelvey, R.D. and Zavoina, W. (1975), "A Statistical Model for the Analysis of Ordinal Level Dependent Variables," *Journal of Mathematical Sociology*, 4, 103–120.

Mroz, T.A. (1987), "The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions," *Econometrica*, 55, 765–799.

Mroz, T.A. (1999), “Discrete Factor Approximations in Simultaneous Equation Models: Estimating the Impact of a Dummy Endogenous Variable on a Continuous Outcome,” *Journal of Econometrics*, 92, 233–274.

Nawata, K. (1994), “Estimation of Sample Selection Bias Models by the Maximum Likelihood Estimator and Heckman’s Two-Step Estimator,” *Economics Letters*, 45, 33–40.

Parks, R.W. (1967), “Efficient Estimation of a System of Regression Equations When Disturbances Are Both Serially and Contemporaneously Correlated,” *Journal of the American Statistical Association*, 62, 500–509.

Powers, D.A. and Xie, Y. (2000), *Statistical Methods for Categorical Data Analysis*, San Diego: Academic Press.

Wooldridge, J.M. (2002), *Econometric Analysis of Cross Section of Panel Data*, Cambridge, MA: MIT Press.

Chapter 23
The SIMLIN Procedure

Chapter Contents

Chapter 23 The SIMLIN Procedure

Overview

The SIMLIN procedure reads the coefficients for a set of linear structural equations, which are usually produced by the SYSLIN procedure. PROC SIMLIN then computes the reduced form and, if input data are given, uses the reduced form equations to generate predicted values. PROC SIMLIN is especially useful when dealing with sets of structural difference equations. The SIMLIN procedure can perform simulation or forecasting of the endogenous variables.

The SIMLIN procedure can be applied only to models that are:

- linear with respect to the parameters
- linear with respect to the variables
- square (as many equations as endogenous variables)
- nonsingular (the coefficients of the endogenous variables form an invertible matrix)

Getting Started

The SIMLIN procedure processes the coefficients in a data set created by the SYSLIN procedure using the OUTEST= option or by another regression procedure such as PROC REG. To use PROC SIMLIN you must first produce the coefficient data set and then specify this data set on the EST= option of the PROC SIMLIN statement. You must also tell PROC SIMLIN which variables are endogenous and which variables are exogenous. List the endogenous variables in an ENDOGENOUS statement, and list the exogenous variables in an EXOGENOUS statement.

The following example illustrates the creation of an OUTEST= data set with PROC SYSLIN and the computation and printing of the reduced form coefficients for the model with PROC SIMLIN.

```
proc syslin data=in outest=e;
   model y1 = y2 x1;
   model y2 = y1 x2;
run;

proc simlin est=e;
   endogenous y1 y2;
   exogenous x1 x2;
run;
```

If the model contains lagged endogenous variables you must also use a LAGGED statement to tell PROC SIMLIN which variables contain lagged values, which endogenous variables they are lags of, and the number of periods of lagging. For dynamic models, the TOTAL and INTERIM= options can be used on the PROC SIMLIN statement to compute and print total and impact multipliers. (See "Dynamic Multipliers" later in this section for an explanation of multipliers.)

In the following example the variables Y1LAG1, Y2LAG1, and Y2LAG2 contain lagged values of the endogenous variables Y1 and Y2. Y1LAG1 and Y2LAG1 contain values of Y1 and Y2 for the previous observation, while Y2LAG2 contains 2 period lags of Y2. The LAGGED statement specifies the lagged relationships, and the TOTAL and INTERIM= options request multiplier analysis. The INTERIM=2 option prints matrices showing the impact that changes to the exogenous variables have on the endogenous variables after 1 and 2 periods.

```
data in; set in;
  y1lag1 = lag(y1);
  y2lag1 = lag(y2);
  y2lag2 = lag2(y2);
run;

proc syslin data=in outest=e;
   model y1 = y2 y1lag1 y2lag2 x1;
   model y2 = y1 y2lag1 x2;
run;

proc simlin est=e total interim=2;
   endogenous y1 y2;
   exogenous x1 x2;
   lagged y1lag1 y1 1 y2lag1 y2 1 y2lag2 y2 2;
run;
```

After the reduced form of the model is computed, the model can be simulated by specifying an input data set on the PROC SIMLIN statement and using an OUTPUT statement to write the simulation results to an output data set. The following example modifies the PROC SIMLIN step from the preceding example to simulate the model and stores the results in an output data set.

```
proc simlin est=e total interim=2 data=in;
   endogenous y1 y2;
   exogenous x1 x2;
   lagged y1lag1 y1 1 y2lag1 y2 1 y2lag2 y2 2;
   output out=sim predicted=y1hat y2hat
                  residual=y1resid y2resid;
run;
```

Prediction and Simulation

If an input data set is specified with the DATA= option in the PROC SIMLIN statement, the procedure reads the data and uses the reduced form equations to compute predicted and residual values for each of the endogenous variables. (If no data set is specified with the DATA= option, no simulation of the system is performed, and only the reduced form and multipliers are computed.)

The character of the prediction is based on the START= value. Until PROC SIMLIN encounters the START= observation, actual endogenous values are found and fed into the lagged endogenous terms. Once the START= observation is reached, dynamic simulation begins, where predicted values are fed into lagged endogenous terms until the end of the data set is reached.

The predicted and residual values generated here are different from those produced by the SYSLIN procedure since PROC SYSLIN uses the structural form with actual endogenous values. The predicted values computed by the SIMLIN procedure solve the simultaneous equation system. These reduced-form predicted values are functions only of the exogenous and lagged endogenous variables and do not depend on actual values of current period endogenous variables.

Syntax

The following statements can be used with PROC SIMLIN:

> **PROC SIMLIN** *options*;
> **BY** *variables*;
> **ENDOGENOUS** *variables*;
> **EXOGENOUS** *variables*;
> **ID** *variables*;
> **LAGGED** *lag-var endogenous-var number ellipsis* **;**
> **OUTPUT** *OUT=SAS-data-set options*;

Functional Summary

The statements and options controlling the SIMLIN procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify input data set containing structural coefficients	PROC SIMLIN	EST=
specify type of estimates read from EST= data set	PROC SIMLIN	TYPE=
write reduced form coefficients and multipliers to an output data set	PROC SIMLIN	OUTEST=
specify the input data set for simulation	PROC SIMLIN	DATA=
write predicted and residual values to an output data set	OUTPUT	
Printing Control Options		
print the structural coefficients	PROC SIMLIN	ESTPRINT
suppress printing of reduced form coefficients	PROC SIMLIN	NORED
suppress all printed output	PROC SIMLIN	NOPRINT
Dynamic Multipliers		
compute interim multipliers	PROC SIMLIN	INTERIM=
compute total multipliers	PROC SIMLIN	TOTAL
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the endogenous variables	ENDOGENOUS	
specify the exogenous variables	EXOGENOUS	
specify identifying variables	ID	
specify lagged endogenous variables	LAGGED	

Description	Statement	Option
Controlling the Simulation		
specify the starting observation for dynamic simulation	PROC SIMLIN	START=

PROC SIMLIN Statement

PROC SIMLIN *options;*

The following options can be used in the PROC SIMLIN statement:

DATA= *SAS-data-set*
specifies the SAS data set containing input data for the simulation. If the DATA= option is used, the data set specified must supply values for all exogenous variables throughout the simulation. If the DATA= option is not specified, no simulation of the system is performed, and only the reduced form and multipliers are computed.

EST= *SAS-data-set*
specifies the input data set containing the structural coefficients of the system. If EST= is omitted the most recently created SAS data set is used. The EST= data set is normally a "TYPE=EST" data set produced by the OUTEST= option of PROC SYSLIN. However, you can also build the EST= data set with a SAS DATA step. See "The EST= Data Set" later in this chapter for details.

ESTPRINT
prints the structural coefficients read from the EST= data set.

INTERIM= *n*
rssbjixinterim multipliersSIMLIN procedure requests that interim multipliers be computed for interims 1 through *n*. If not specified, no interim multipliers are computed. This feature is available only if there are no lags greater than 1.

NOPRINT
suppresses all printed output.

NORED
suppresses the printing of the reduced form coefficients.

OUTEST= *SAS-data-set*
specifies an output SAS data set to contain the reduced form coefficients and multipliers, in addition to the structural coefficients read from the EST= data set. The OUTEST= data set has the same form as the EST= data set. If the OUTEST= option is not specified, the reduced form coefficients and multipliers are not written to a data set.

START= *n*

specifies the observation number in the DATA= data set where the dynamic simulation is to be started. By default, the dynamic simulation starts with the first observation in the DATA= data set for which all variables (including lags) are not missing.

TOTAL

requests that the total multipliers be computed. This feature is available only if there are no lags greater than 1.

TYPE= *value*

specifies the type of estimates to be read from the EST= data set. The TYPE= value must match the value of the _TYPE_ variable for the observations that you want to select from the EST= data set (TYPE=2SLS, for example).

BY Statement

BY *variables;*

A BY statement can be used with PROC SIMLIN to obtain separate analyses for groups of observations defined by the BY variables.

The BY statement can be applied to one or both of the EST= and the DATA= input data set. When a BY statement is used and both an EST= and a DATA= input data set are specified, PROC SIMLIN checks to see if one or both of the data sets contain the BY variables.

Thus, there are three ways of using the BY statement with PROC SIMLIN:

1. If the BY variables are found in the EST= data set only, PROC SIMLIN simulates over the entire DATA= data set once for each set of coefficients read from the BY groups in the EST= data set.
2. If the BY variables are found in the DATA= data set only, PROC SIMLIN performs separate simulations over each BY group in the DATA= data set, using the single set of coefficients in the EST= data set.
3. If the BY variables are found in both the EST= and the DATA= data sets, PROC SIMLIN performs separate simulations over each BY group in the DATA= data set using the coefficients from the corresponding BY group in the EST= data set.

ENDOGENOUS Statement

ENDOGENOUS *variables;*

List the names of the endogenous (jointly dependent) variables in the ENDOGENOUS statement. The ENDOGENOUS statement can be abbreviated as ENDOG or ENDO.

EXOGENOUS Statement

EXOGENOUS *variables;*

List the names of the exogenous (independent) variables in the EXOGENOUS statement. The EXOGENOUS statement can be abbreviated as EXOG or EXO.

ID Statement

ID *variables;*

The ID statement can be used to restrict the variables copied from the DATA= data set to the OUT= data set. Use the ID statement to list the variables you want copied to the OUT= data set besides the exogenous, endogenous, lagged endogenous, and BY variables. If the ID statement is omitted, all the variables in the DATA= data set are copied to the OUT= data set.

LAGGED Statement

LAGGED *lag-var endogenous-var number ellipsis ;*

For each lagged endogenous variable, specify the name of the lagged variable, the name of the endogenous variable that was lagged, and the degree of the lag. Only one LAGGED statement is allowed.

The following is an example of the use of the LAGGED statement:

```
proc simlin est=e;
   endog y1 y2;
   lagged y1lag1 y1 1  y2lag1 y2 1  y2lag3 y2 3;
```

This statement specifies that the variable Y1LAG1 contains the values of the endogenous variable Y1 lagged one period; the variable Y2LAG1 refers to the values of Y2 lagged one period; and the variable Y2LAG3 refers to the values of Y2 lagged three periods.

OUTPUT Statement

OUTPUT *OUT= SAS-data-set options;*

The OUTPUT statement specifies that predicted and residual values be put in an output data set. A DATA= input data set must be supplied if the OUTPUT statement is used, and only one OUTPUT statement is allowed. The following options can be used in the OUTPUT statement:

OUT= *SAS-data-set*
names the output SAS data set to contain the predicted values and residuals. If OUT= is not specified, the output data set is named using the DATA*n* convention.

PREDICTED= *names*
P= *names*
names the variables in the output data set that contain the predicted values of the simulation. These variables correspond to the endogenous variables in the order in which they are specified in the ENDOGENOUS statement. Specify up to as many names as there are endogenous variables. If you specify names on the PREDICTED= option for only some of the endogenous variables, predicted values for the remaining variables are not output. The names must not match any variable name in the input data set.

RESIDUAL= *names*
R= *names*
names the variables in the output data set that contain the residual values from the simulation. The residuals are the differences between the actual values of the endogenous variables from the DATA= data set and the predicted values from the simulation. These variables correspond to the endogenous variables in the order in which they are specified in the ENDOGENOUS statement. Specify up to as many names as there are endogenous variables. The names must not match any variable name in the input data set.

The following is an example of the use of the OUTPUT statement. This example outputs predicted values for Y1 and Y2 and outputs residuals for Y1.

```
proc simlin est=e;
   endog y1 y2;
   output out=b predicted=y1hat y2hat residual=y1resid;
```

Details

The following sections explain the structural and reduced forms, dynamic multipliers, input data sets, and the model simulation process in more detail.

Defining the Structural Form

An EST= input data set supplies the coefficients of the equation system. The data set containing the coefficients is normally a "TYPE=EST" data set created by the OUTEST= option of PROC SYSLIN or another regression procedure. The data set contains the special variables _TYPE_, _DEPVAR_, and INTERCEPT. You can also supply the structural coefficients of the system to PROC SIMLIN in a data set produced by a SAS DATA step as long as the data set is of the form TYPE=EST. Refer to SAS/STAT software documentation for a discussion of the special TYPE=EST type of SAS data set.

Suppose that there is a $g \times 1$ vector of endogenous variables $\mathbf{y}_t$, an $l \times 1$ vector of lagged endogenous variables $\mathbf{y}_t^L$, and a $k \times 1$ vector of exogenous variables $\mathbf{x}_t$, including the intercept. Then, there are g structural equations in the simultaneous system that can be written

$$\mathbf{G}\mathbf{y}_t = \mathbf{C}\mathbf{y}_t^L + \mathbf{B}\mathbf{x}_t$$

where $\mathbf{G}$ is the matrix of coefficients of current period endogenous variables, $\mathbf{C}$ is the matrix of coefficients of lagged endogenous variables, and $\mathbf{B}$ is the matrix of coefficients of exogenous variables. $\mathbf{G}$ is assumed to be nonsingular.

Computing the Reduced Form

First, the SIMLIN procedure computes reduced form coefficients by premultiplying by $\mathbf{G}^{-1}$:

$$\mathbf{y}_t = \mathbf{G}^{-1}\mathbf{C}\mathbf{y}_t^L + \mathbf{G}^{-1}\mathbf{B}\mathbf{x}_t$$

This can be written as

$$\mathbf{y}_t = \Pi_1\mathbf{y}_t^L + \Pi_2\mathbf{x}_t$$

where Π_1= $\mathbf{G}^{-1}\mathbf{C}$ and Π_2= $\mathbf{G}^{-1}\mathbf{B}$ are the reduced form coefficient matrices.

The reduced form matrices Π_1= $\mathbf{G}^{-1}\mathbf{C}$ and Π_2= $\mathbf{G}^{-1}\mathbf{B}$ are printed unless the NORED option is specified in the PROC SIMLIN statement. The structural coefficient matrices $\mathbf{G}$, $\mathbf{C}$, and $\mathbf{B}$ are printed when the ESTPRINT option is specified.

Dynamic Multipliers

For models that have only first-order lags, the equation of the reduced form of the system can be rewritten

$$\mathbf{y}_t = \mathbf{D}\mathbf{y}_{t-1} + \Pi_2 \mathbf{x}_t$$

D is a matrix formed from the columns of Π_1 plus some columns of zeros, arranged in the order in which the variables meet the lags. The elements of Π_2 are called *impact multipliers* because they show the immediate effect of changes in each exogenous variable on the values of the endogenous variables. This equation can be rewritten as

$$\mathbf{y}_t = \mathbf{D}^2\mathbf{y}_{t-2} + \mathbf{D}\Pi_2\mathbf{x}_{t-1} + \Pi_2 \mathbf{x}_t$$

The matrix formed by the product $\mathbf{D}\Pi_2$ shows the effect of the exogenous variables one lag back; the elements in this matrix are called *interim multipliers* and are computed and printed when the INTERIM= option is specified in the PROC SIMLIN statement. The *i*th period interim multipliers are formed by $\mathbf{D}^i\Pi_2$.

The series can be expanded as

$$\mathbf{y}_t = \mathbf{D}^{\infty}\mathbf{y}_{t-\infty} + \sum_{i=0}^{\infty} \mathbf{D}^i \Pi_2 \mathbf{x}_{t-i}$$

A permanent and constant setting of a value for *x* has the following cumulative effect:

$$\left(\sum_{i=0}^{\infty} \mathbf{D}^i \right) \Pi_2 \mathbf{x} = (\mathbf{I} - \mathbf{D})^{-1} \Pi_2 \mathbf{x}$$

The elements of $(\mathbf{I}\text{-}\mathbf{D})^{-1}\Pi_2$ are called the *total multipliers*. Assuming that the sum converges and that (**I-D**) is invertible, PROC SIMLIN computes the total multipliers when the TOTAL option is specified in the PROC SIMLIN statement.

Multipliers for Higher Order Lags

The dynamic multiplier options require the system to have no lags of order greater than one. This limitation can be circumvented, since any system with lags greater than one can be rewritten as a system where no lag is greater than one by forming new endogenous variables that are single-period lags.

For example, suppose you have the third-order single equation

$$y_t = a y_{t-3} + b\mathbf{x}_t$$

This can be converted to a first-order three-equation system by introducing two additional endogenous variables, $y_{1,t}$ and $y_{2,t}$, and computing corresponding first-order lagged variables for each endogenous variable: y_{t-1}, $y_{1,t-1}$, and $y_{2,t-1}$. The higher order lag relations are then produced by adding identities to link the endogenous and identical lagged endogenous variables:

$$y_{1,t} = y_{t-1}$$

$$y_{2,t} = y_{1,t-1}$$

$$y_t = ay_{2,t-1} + b\mathbf{X}_t$$

This conversion using the SYSLIN and SIMLIN procedures requires three steps:

1. Add the extra endogenous and lagged endogenous variables to the input data set using a DATA step. Note that two copies of each lagged endogenous variable are needed for each lag reduced, one to serve as an endogenous variable and one to serve as a lagged endogenous variable in the reduced system.
2. Add IDENTITY statements to the PROC SYSLIN step to equate each added endogenous variable to its lagged endogenous variable copy.
3. In the PROC SIMLIN step, declare the added endogenous variables in the ENDOGENOUS statement and define the lag relations in the LAGGED statement.

See Example 23.2 for an illustration of how to convert an equation system with higher-order lags into a larger system with only first-order lags.

EST= Data Set

Normally, PROC SIMLIN uses an EST= data set produced by PROC SYSLIN with the OUTEST= option. This data set is in the form expected by PROC SIMLIN. If there is more than one set of estimates produced by PROC SYSLIN, you must use the TYPE= option in the PROC SIMLIN statement to select the set to be simulated. Then PROC SIMLIN reads from the EST= data set only those observations with a _TYPE_ value corresponding to the TYPE= option (for example, TYPE=2SLS) or with a _TYPE_ value of IDENTITY.

The SIMLIN procedure can only solve square, nonsingular systems. If you have fewer equations than endogenous variables, you must specify IDENTITY statements in the PROC SYSLIN step to bring the system up to full rank. If there are *g* endogenous variables and *m*<*g* stochastic equations with unknown parameters, then you use *m* MODEL statements to specify the equations with parameters to be estimated and you must use *g*-*m* IDENTITY statements to complete the system.

You can build your own EST= data set with a DATA step rather than use PROC SYSLIN. The EST= data set must contain the endogenous variables, the lagged endogenous variables (if any), and the exogenous variables in the system (if any). If any of the equations have intercept terms, the variable INTERCEPT must supply these coefficients. The EST= data set should also contain the special character variable comp _DEPVAR_ to label the equations.

The EST= data set must contain one observation for each equation in the system. The values of the lagged endogenous variables must contain the **C** coefficients. The values of the exogenous variables and the INTERCEPT variable must contain the **B** coefficients. The values of the endogenous variables, however, must contain the negatives of the **G** coefficients. This is because the SYSLIN procedure writes the coefficients to the OUTEST= data set in the form

$$0 = \mathbf{H}\mathbf{y}_t + \mathbf{C}\mathbf{y}_t^L + \mathbf{B}\mathbf{x}_t$$

where **H=-G**.

See "Multipliers for Higher Order Lags" and Example 23.2 later in this chapter for more information on building the EST= data set.

DATA= Data Set

The DATA= data set must contain all of the exogenous variables. Values for all of the exogenous variables are required for each observation for which predicted endogenous values are desired. To forecast past the end of the historical data, the DATA= data set should contain nonmissing values for all of the exogenous variables and missing values for the endogenous variables for the forecast periods, in addition to the historical data. (See Example 23.1 for an illustration.)

In order for PROC SIMLIN to output residuals and compute statistics of fit, the DATA= data set must also contain the endogenous variables with nonmissing actual values for each observation for which residuals and statistics are to be computed.

If the system contains lags, initial values must be supplied for the lagged variables. This can be done by including either the lagged variables or the endogenous variables, or both, in the DATA= data set. If the lagged variables are not in the DATA= data set or if they have missing values in the early observations, PROC SIMLIN prints a warning and uses the endogenous variable values from the early observations to initialize the lags.

OUTEST= Data Set

The OUTEST= data set contains all the variables read from the EST= data set. The variables in the OUTEST= data set are as follows.

- the BY statement variables, if any
- _TYPE_, a character variable that identifies the type of observation
- _DEPVAR_, a character variable containing the name of the dependent variable for the observation
- the endogenous variables

- the lagged endogenous variables
- the exogenous variables
- INTERCEPT, a numeric variable containing the intercept values
- _MODEL_, a character variable containing the name of the equation
- _SIGMA_, a numeric variable containing the estimated error variance of the equation (output only if present in the EST= data set)

The observations read from the EST= data set that supply the structural coefficients are copied to the OUTEST= data set, except that the signs of endogenous coefficients are reversed. For these observations, the _TYPE_ variable values are the same as in the EST= data set.

In addition, the OUTEST= data set contains observations with the following _TYPE_ values:

REDUCED	the reduced form coefficients. The endogenous variables for this group of observations contain the inverse of the endogenous coefficient matrix **G**. The lagged endogenous variables contain the matrix $\Pi_1=\mathbf{G}^{-1}\mathbf{C}$. The exogenous variables contain the matrix $\Pi_2=\mathbf{G}^{-1}\mathbf{B}$.
IMULT*i*	the interim multipliers, if the INTERIM= option is specified. There are *gn* observations for the interim multipliers, where *g* is the number of endogenous variables and *n* is the value of the INTERIM=*n* option. For these observations the _TYPE_ variable has the value IMULT*i*, where the interim number *i* ranges from 1 to *n*. The exogenous variables in groups of *g* observations that have a _TYPE_ value of IMULT*i* contain the matrix $\mathbf{D}^i\Pi_2$ of multipliers at interim *i*. The endogenous and lagged endogenous variables for this group of observations are set to missing.
TOTAL	the total multipliers, if the TOTAL option is specified. The exogenous variables in this group of observations contain the matrix $(\mathbf{I}\text{-}\mathbf{D})^{-1}\Pi_2$. The endogenous and lagged endogenous variables for this group of observations are set to missing.

OUT= Data Set

The OUT= data set normally contains all of the variables in the input DATA= data set, plus the variables named in the PREDICTED= and RESIDUAL= options in the OUTPUT statement.

You can use an ID statement to restrict the variables that are copied from the input data set. If an ID statement is used, the OUT= data set contains only the BY variables (if any), the ID variables, the endogenous and lagged endogenous variables (if any), the exogenous variables, plus the PREDICTED= and RESIDUAL= variables.

The OUT= data set contains an observation for each observation in the DATA= data set. When the actual value of an endogenous variable is missing in the DATA= data set, or when the DATA= data set does not contain the endogenous variable, the corresponding residual is missing.

Printed Output

Structural Form

The following items are printed as they are read from the EST= input data set. Structural zeros are printed as dots in the listing of these matrices.

1. Structural Coefficients for Endogenous Variables. This is the **G** matrix, with g rows and g columns.
2. Structural Coefficients for Lagged Endogenous Variables. These coefficients make up the **C** matrix, with g rows and l columns.
3. Structural Coefficients for Exogenous Variables. These coefficients make up the **B** matrix, with g rows and k columns.

Reduced Form

1. The reduced form coefficients are obtained by inverting **G** so that the endogenous variables can be directly expressed as functions of only lagged endogenous and exogenous variables.
2. Inverse Coefficient Matrix for Endogenous Variables. This is the inverse of the **G** matrix.
3. Reduced Form for Lagged Endogenous Variables. This is $\Pi_1 = \mathbf{G}^{-1}\mathbf{C}$, with g rows and l columns. Each value is a dynamic multiplier that shows how past values of lagged endogenous variables affect values of each of the endogenous variables.
4. Reduced Form for Exogenous Variables. This is $\Pi_2 = \mathbf{G}^{-1}\mathbf{B}$, with g rows and k columns. Its values are called *impact multipliers* because they show the immediate effect of each exogenous variable on the value of the endogenous variables.

Multipliers

Interim and total multipliers show the effect of a change in an exogenous variable over time.

1. Interim Multipliers. These are the interim multiplier matrices. They are formed by multiplying Π_2 by powers of **D**. The dth interim multiplier is $\mathbf{D}^d\Pi_2$. The interim multiplier of order d shows the effects of a change in the exogenous variables after d periods. Interim multipliers are only available if the maximum lag of the endogenous variables is 1.
2. Total Multipliers. This is the matrix of total multipliers, $T=(\mathbf{I}\text{-}\mathbf{D})^{-1}\Pi_2$. This matrix shows the cumulative effect of changes in the exogenous variables. Total multipliers are only available if the maximum lag is one.

Statistics of Fit

If the DATA= option is used and the DATA= data set contains endogenous variables, PROC SIMLIN prints a statistics-of-fit report for the simulation. The statistics printed include the following. (Summations are over the observations for which both y_t and $\hat{y}_t$ are nonmissing.)

1. the number of nonmissing errors. (Number of observations for which both y_t and $\hat{y}_t$ are nonmissing.)
2. the mean error: $\frac{1}{n}\sum(y_t - \hat{y}_t)$
3. the mean percent error: $\frac{100}{n}\sum\frac{(y_t-\hat{y}_t)}{y_t}$
4. the mean absolute error: $\frac{1}{n}\sum|y_t - \hat{y}_t|$
5. the mean absolute percent error $\frac{100}{n}\sum\frac{|y_t-\hat{y}_t|}{y_t}$
6. the root mean square error: $\sqrt{\frac{1}{n}\sum(y_t - \hat{y}_t)^2}$
7. the root mean square percent error: $\sqrt{\frac{100}{n}\sum(\frac{(y_t-\hat{y}_t)}{y_t})^2}$

ODS Table Names

PROC SIMLIN assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 23.1. ODS Tables Produced in PROC SIMLIN

ODS Table Name	**Description**	**Option**
Endogenous	Structural Coefficients for Endogenous Variables	default
LaggedEndogenous	Structural Coefficients for Lagged Endogenous Variables	default
Exogenous Structural	Coefficients for Exogenous Variables	default
InverseCoeff	Inverse Coefficient Matrix for Endogenous Variables	default
RedFormLagEndo	Reduced Form for Lagged Endogenous Variables	default
RedFormExog	Reduced Form for Exogenous Variables	default
InterimMult	Interim Multipliers	INTERIM= option
TotalMult	Total Multipliers	TOTAL= option
FitStatistics	Fit statistics	default

Examples

Example 23.1. Simulating Klein’s Model I

In this example, the SIMLIN procedure simulates a model of the U.S. economy called Klein’s Model I. The SAS data set KLEIN, shown in Output 23.1.1, is used as input to the SYSLIN and SIMLIN procedures.

```
data klein;
   input year c p w i x wp g t k wsum;
   date=mdy(1,1,year);
   format date year.;
   y   =c+i+g-t;
   yr  =year-1931;
   klag=lag(k);
   plag=lag(p);
   xlag=lag(x);
   if year>=1921;
   label c   ='consumption'
         p   ='profits'
         w   ='private wage bill'
         i   ='investment'
         k   ='capital stock'
         y   ='national income'
         x   ='private production'
         wsum='total wage bill'
         wp  ='govt wage bill'
         g   ='govt demand'
         t   ='taxes'
         klag='capital stock lagged'
         plag='profits lagged'
         xlag='private product lagged'
         yr  ='year-1931';
   datalines;
  ... data lines omitted ...
proc print data=klein;
run;
```

Output 23.1.1. PROC PRINT Listing of Input Data Set KLEIN

```
 Obs  year    c     p     w     i     x    wp     g     t     k    wsum  date    y    yr   klag  plag  xlag

   1  1921  41.9  12.4  25.5  -0.2  45.6  2.7   3.9   7.7  182.6  28.2  1921  37.9  -10  182.8  12.7  44.9
   2  1922  45.0  16.9  29.3   1.9  50.1  2.9   3.2   3.9  184.5  32.2  1922  46.2   -9  182.6  12.4  45.6
   3  1923  49.2  18.4  34.1   5.2  57.2  2.9   2.8   4.7  189.7  37.0  1923  52.5   -8  184.5  16.9  50.1
   4  1924  50.6  19.4  33.9   3.0  57.1  3.1   3.5   3.8  192.7  37.0  1924  53.3   -7  189.7  18.4  57.2
   5  1925  52.6  20.1  35.4   5.1  61.0  3.2   3.3   5.5  197.8  38.6  1925  55.5   -6  192.7  19.4  57.1
   6  1926  55.1  19.6  37.4   5.6  64.0  3.3   3.3   7.0  203.4  40.7  1926  57.0   -5  197.8  20.1  61.0
   7  1927  56.2  19.8  37.9   4.2  64.4  3.6   4.0   6.7  207.6  41.5  1927  57.7   -4  203.4  19.6  64.0
   8  1928  57.3  21.1  39.2   3.0  64.5  3.7   4.2   4.2  210.6  42.9  1928  60.3   -3  207.6  19.8  64.4
   9  1929  57.8  21.7  41.3   5.1  67.0  4.0   4.1   4.0  215.7  45.3  1929  63.0   -2  210.6  21.1  64.5
  10  1930  55.0  15.6  37.9   1.0  61.2  4.2   5.2   7.7  216.7  42.1  1930  53.5   -1  215.7  21.7  67.0
  11  1931  50.9  11.4  34.5  -3.4  53.4  4.8   5.9   7.5  213.3  39.3  1931  45.9    0  216.7  15.6  61.2
  12  1932  45.6   7.0  29.0  -6.2  44.3  5.3   4.9   8.3  207.1  34.3  1932  36.0    1  213.3  11.4  53.4
  13  1933  46.5  11.2  28.5  -5.1  45.1  5.6   3.7   5.4  202.0  34.1  1933  39.7    2  207.1   7.0  44.3
  14  1934  48.7  12.3  30.6  -3.0  49.7  6.0   4.0   6.8  199.0  36.6  1934  42.9    3  202.0  11.2  45.1
  15  1935  51.3  14.0  33.2  -1.3  54.4  6.1   4.4   7.2  197.7  39.3  1935  47.2    4  199.0  12.3  49.7
  16  1936  57.7  17.6  36.8   2.1  62.7  7.4   2.9   8.3  199.8  44.2  1936  54.4    5  197.7  14.0  54.4
  17  1937  58.7  17.3  41.0   2.0  65.0  6.7   4.3   6.7  201.8  47.7  1937  58.3    6  199.8  17.6  62.7
  18  1938  57.5  15.3  38.2  -1.9  60.9  7.7   5.3   7.4  199.9  45.9  1938  53.5    7  201.8  17.3  65.0
  19  1939  61.6  19.0  41.6   1.3  69.5  7.8   6.6   8.9  201.2  49.4  1939  60.6    8  199.9  15.3  60.9
  20  1940  65.0  21.1  45.0   3.3  75.7  8.0   7.4   9.6  204.5  53.0  1940  66.1    9  201.2  19.0  69.5
  21  1941  69.7  23.5  53.3   4.9  88.4  8.5  13.8  11.6  209.4  61.8  1941  76.8   10  204.5  21.1  75.7
  22  1942    .     .     .     .     .   8.5  13.8  11.6     .     .   1942    .    11  209.4  23.5  88.4
  23  1943    .     .     .     .     .   8.5  13.8  12.6     .     .   1943    .    12     .     .     .
  24  1944    .     .     .     .     .   8.5  13.8  11.6     .     .   1944    .    13     .     .     .
  25  1945    .     .     .     .     .   8.5  13.8  11.6     .     .   1945    .    14     .     .     .
  26  1946    .     .     .     .     .   8.5  13.8  11.6     .     .   1946    .    15     .     .     .
  27  1947    .     .     .     .     .   8.5  13.8  11.6     .     .   1947    .    16     .     .     .
```

First, the model is specified and estimated using the SYSLIN procedure, and the parameter estimates are written to an OUTEST= data set. The printed output produced by the SYSLIN procedure is not shown here; see Example 26.1 in Chapter 26 for the printed output of the PROC SYSLIN step.

```
title1 'Simulation of Klein''s Model I using SIMLIN';
proc syslin 3sls data=klein outest=a;

   instruments klag plag xlag wp g t yr;
   endogenous c p w i x wsum k y;

   consume: model    c = p plag wsum;
   invest:  model    i = p plag klag;
   labor:   model    w = x xlag yr;

   product: identity x = c + i + g;
   income:  identity y = c + i + g - t;
   profit:  identity p = x - w - t;
   stock:   identity k = klag + i;
   wage:    identity wsum = w + wp;
run;

proc print data=a;
run;
```

The OUTEST= data set A created by the SYSLIN procedure contains parameter estimates to be used by the SIMLIN procedure. The OUTEST= data set is shown in Output 23.1.2.

Output 23.1.2. The OUTEST= Data Set Created by PROC SYSLIN

```
                                    Simulation of Klein's Model I using SIMLIN

Obs   _TYPE_       _STATUS_      _MODEL_    _DEPVAR_    _SIGMA_    Intercept      klag        plag       xlag

  1   INST        0 Converged    FIRST        c         2.11403     58.3018     -0.14654    0.74803     0.23007
  2   INST        0 Converged    FIRST        p         2.18298     50.3844     -0.21610    0.80250     0.02200
  3   INST        0 Converged    FIRST        w         1.75427     43.4356     -0.12295    0.87192     0.09533
  4   INST        0 Converged    FIRST        i         1.72376     35.5182     -0.19251    0.92639    -0.11274
  5   INST        0 Converged    FIRST        x         3.77347     93.8200     -0.33906    1.67442     0.11733
  6   INST        0 Converged    FIRST        wsum      1.75427     43.4356     -0.12295    0.87192     0.09533
  7   INST        0 Converged    FIRST        k         1.72376     35.5182      0.80749    0.92639    -0.11274
  8   INST        0 Converged    FIRST        y         3.77347     93.8200     -0.33906    1.67442     0.11733
  9   3SLS        0 Converged    CONSUME      c         1.04956     16.4408        .        0.16314       .
 10   3SLS        0 Converged    INVEST       i         1.60796     28.1778     -0.19485    0.75572       .
 11   3SLS        0 Converged    LABOR        w         0.80149      1.7972        .          .         0.18129
 12   IDENTITY    0 Converged    PRODUCT      x           .           0.0000        .          .           .
 13   IDENTITY    0 Converged    INCOME       y           .           0.0000        .          .           .
 14   IDENTITY    0 Converged    PROFIT       p           .           0.0000        .          .           .
 15   IDENTITY    0 Converged    STOCK        k           .           0.0000      1.00000      .           .
 16   IDENTITY    0 Converged    WAGE         wsum        .           0.0000        .          .           .

Obs      wp          g           t          yr       c        p        w     i        x         wsum        k     y

  1    0.19327    0.20501    -0.36573    0.70109    -1      .          .     .      .           .           .     .
  2   -0.07961    0.43902    -0.92310    0.31941     .    -1.00000     .     .      .           .           .     .
  3   -0.44373    0.86622    -0.60415    0.71358     .      .         -1     .      .           .           .     .
  4   -0.71661    0.10023    -0.16152    0.33190     .      .          .    -1      .           .           .     .
  5   -0.52334    1.30524    -0.52725    1.03299     .      .          .     .    -1.00000      .           .     .
  6    0.55627    0.86622    -0.60415    0.71358     .      .          .     .      .         -1.00000      .     .
  7   -0.71661    0.10023    -0.16152    0.33190     .      .          .     .      .           .          -1     .
  8   -0.52334    1.30524    -1.52725    1.03299     .      .          .     .      .           .           .    -1
  9     .          .           .          .         -1     0.12489     .     .      .          0.79008      .     .
 10     .          .           .          .          .    -0.01308     .    -1      .           .           .     .
 11     .          .           .         0.14967     .      .         -1     .     0.40049      .           .     .
 12     .         1.00000      .          .          1      .          .     1    -1.00000      .           .     .
 13     .         1.00000    -1.00000     .          1      .          .     1      .           .           .    -1
 14     .          .         -1.00000     .          .    -1.00000    -1     .     1.00000      .           .     .
 15     .          .           .          .          .      .          .     1      .           .          -1     .
 16    1.00000     .           .          .          .      .          1     .      .         -1.00000      .     .
```

Using the OUTEST= data set A produced by the SYSLIN procedure, the SIMLIN procedure can now compute the reduced form and simulate the model. The following statements perform the simulation.

```
proc simlin est=a data=klein type=3sls
            estprint total interim=2 outest=b;
   endogenous c p w i x wsum k y;
   exogenous  wp g t yr;
   lagged  klag k 1   plag p 1   xlag x 1;
   id year;
   output out=c p=chat phat what ihat xhat wsumhat khat yhat
                r=cres pres wres ires xres wsumres kres yres;
run;
```

The reduced form coefficients and multipliers are added to the information read from EST= data set A and written to the OUTEST= data set B. The predicted and residual values from the simulation are written to the OUT= data set C specified in the OUTPUT statement.

The SIMLIN procedure first prints the structural coefficient matrices read from the EST= data set, as shown in Output 23.1.3.

Output 23.1.3. SIMLIN Procedure Output – Structural Coefficients

```
                    Simulation of Klein's Model I using SIMLIN

                              The SIMLIN Procedure

                 Structural Coefficients for Endogenous Variables

      Variable                c              p              w              i

      c                  1.0000        -0.1249              .              .
      i                       .         0.0131              .         1.0000
      w                       .              .         1.0000              .
      x                 -1.0000              .              .        -1.0000
      y                 -1.0000              .              .        -1.0000
      p                       .         1.0000         1.0000              .
      k                       .              .              .        -1.0000
      wsum                    .              .        -1.0000              .

                 Structural Coefficients for Endogenous Variables

      Variable                x           wsum              k              y

      c                       .        -0.7901              .              .
      i                       .              .              .              .
      w                 -0.4005              .              .              .
      x                  1.0000              .              .              .
      y                       .              .              .         1.0000
      p                 -1.0000              .              .              .
      k                       .              .         1.0000              .
      wsum                    .         1.0000              .              .
```

```
                    Simulation of Klein's Model I using SIMLIN

                              The SIMLIN Procedure

              Structural Coefficients for Lagged Endogenous Variables

             Variable             klag           plag           xlag

             c                       .         0.1631              .
             i                 -0.1948         0.7557              .
             w                       .              .         0.1813
             x                       .              .              .
             y                       .              .              .
             p                       .              .              .
             k                  1.0000              .              .
             wsum                    .              .              .


                 Structural Coefficients for Exogenous Variables

 Variable               wp              g              t             yr      Intercept

 c                       .              .              .              .        16.4408
 i                       .              .              .              .        28.1778
 w                       .              .              .         0.1497         1.7972
 x                       .         1.0000              .              .              0
 y                       .         1.0000        -1.0000              .              0
 p                       .              .        -1.0000              .              0
 k                       .              .              .              .              0
 wsum               1.0000              .              .              .              0
```

The SIMLIN procedure then prints the inverse of the endogenous variables coefficient matrix, as shown in Output 23.1.4.

Output 23.1.4. SIMLIN Procedure Output – Inverse Coefficient Matrix

```
                     Simulation of Klein's Model I using SIMLIN

                                The SIMLIN Procedure

                 Inverse Coefficient Matrix for Endogenous Variables

       Variable              c              i              w              x

       c                1.6347         0.6347         1.0957         0.6347
       p                0.9724         0.9724        -0.3405         0.9724
       w                0.6496         0.6496         1.4406         0.6496
       i               -0.0127         0.9873       0.004453        -0.0127
       x                1.6219         1.6219         1.1001         1.6219
       wsum             0.6496         0.6496         1.4406         0.6496
       k               -0.0127         0.9873       0.004453        -0.0127
       y                1.6219         1.6219         1.1001         0.6219

                 Inverse Coefficient Matrix for Endogenous Variables

       Variable              y              p              k           wsum

       c                     0         0.1959              0         1.2915
       p                     0         1.1087              0         0.7682
       w                     0         0.0726              0         0.5132
       i                     0        -0.0145              0        -0.0100
       x                     0         0.1814              0         1.2815
       wsum                  0         0.0726              0         1.5132
       k                     0        -0.0145         1.0000        -0.0100
       y                1.0000         0.1814              0         1.2815
```

The SIMLIN procedure next prints the reduced form coefficient matrices, as shown in Output 23.1.5.

Output 23.1.5. SIMLIN Procedure Output – Reduced Form Coefficients

```
                    Simulation of Klein's Model I using SIMLIN

                              The SIMLIN Procedure

                  Reduced Form for Lagged Endogenous Variables

            Variable              klag           plag           xlag

            c                  -0.1237         0.7463         0.1986
            p                  -0.1895         0.8935        -0.0617
            w                  -0.1266         0.5969         0.2612
            i                  -0.1924         0.7440       0.000807
            x                  -0.3160         1.4903         0.1994
            wsum               -0.1266         0.5969         0.2612
            k                   0.8076         0.7440       0.000807
            y                  -0.3160         1.4903         0.1994


                       Reduced Form for Exogenous Variables

Variable               wp             g              t             yr      Intercept

c                  1.2915        0.6347        -0.1959         0.1640        46.7273
p                  0.7682        0.9724        -1.1087        -0.0510        42.7736
w                  0.5132        0.6496        -0.0726         0.2156        31.5721
i                 -0.0100       -0.0127         0.0145       0.000667        27.6184
x                  1.2815        1.6219        -0.1814         0.1647        74.3457
wsum               1.5132        0.6496        -0.0726         0.2156        31.5721
k                 -0.0100       -0.0127         0.0145       0.000667        27.6184
y                  1.2815        1.6219        -1.1814         0.1647        74.3457
```

The multiplier matrices (requested by the INTERIM=2 and TOTAL options) are printed next, as shown in Output 23.1.6.

Output 23.1.6. SIMLIN Procedure Output – Multipliers

```
                     Simulation of Klein's Model I using SIMLIN

                                The SIMLIN Procedure

                          Interim Multipliers for Interim 1

Variable             wp             g             t            yr     Intercept

c              0.829130      1.049424     -0.865262     -.0054080      43.27442
p              0.609213      0.771077     -0.982167     -.0558215      28.39545
w              0.794488      1.005578     -0.710961     0.0125018      41.45124
i              0.574572      0.727231     -0.827867     -.0379117      26.57227
x              1.403702      1.776655     -1.693129     -.0433197      69.84670
wsum           0.794488      1.005578     -0.710961     0.0125018      41.45124
k              0.564524      0.714514     -0.813366     -.0372452      54.19068
y              1.403702      1.776655     -1.693129     -.0433197      69.84670


                          Interim Multipliers for Interim 2

Variable             wp             g             t            yr     Intercept

c              0.663671      0.840004     -0.968727     -.0456589      28.36428
p              0.350716      0.443899     -0.618929     -.0401446      10.79216
w              0.658769      0.833799     -0.925467     -.0399178      28.33114
i              0.345813      0.437694     -0.575669     -.0344035      10.75901
x              1.009485      1.277698     -1.544396     -.0800624      39.12330
wsum           0.658769      0.833799     -0.925467     -.0399178      28.33114
k              0.910337      1.152208     -1.389035     -.0716486      64.94969
y              1.009485      1.277698     -1.544396     -.0800624      39.12330
```

```
                     Simulation of Klein's Model I using SIMLIN

                                The SIMLIN Procedure

                                 Total Multipliers

Variable             wp             g             t            yr     Intercept

c              1.881667      1.381613     -0.685987     0.1789624       41.3045
p              0.786945      0.996031     -1.286891     -.0748290       15.4770
w              1.094722      1.385582     -0.399095     0.2537914       25.8275
i              0.000000      0.000000     -0.000000     0.0000000        0.0000
x              1.881667      2.381613     -0.685987     0.1789624       41.3045
wsum           2.094722      1.385582     -0.399095     0.2537914       25.8275
k              2.999365      3.796275     -4.904859     -.2852032      203.6035
y              1.881667      2.381613     -1.685987     0.1789624       41.3045
```

The last part of the SIMLIN procedure output is a table of statistics of fit for the simulation, as shown in Output 23.1.7.

Output 23.1.7. SIMLIN Procedure Output – Simulation Statistics

```
                    Simulation of Klein's Model I using SIMLIN

                               The SIMLIN Procedure

                                  Fit Statistics

                              Mean   Mean Pct   Mean Abs    Mean Abs        RMS    RMS Pct
Variable         N           Error      Error      Error   Pct Error      Error      Error

c               21          0.1367    -0.3827     3.5011     6.69769     4.3155     8.1701
p               21          0.1422    -4.0671     2.9355    19.61400     3.4257    26.0265
w               21          0.1282    -0.8939     3.1247     8.92110     4.0930    11.4709
i               21          0.1337   105.8529     2.4983   127.13736     2.9980   252.3497
x               21          0.2704    -0.9553     5.9622    10.40057     7.1881    12.5653
wsum            21          0.1282    -0.6669     3.1247     7.88988     4.0930    10.1724
k               21         -0.1424    -0.1506     3.8879     1.90614     5.0036     2.4209
y               21          0.2704    -1.3476     5.9622    11.74177     7.1881    14.2214
```

The OUTEST= output data set contains all the observations read from the EST= data set, and in addition contains observations for the reduced form and multiplier matrices. The following statements produce a partial listing of the OUTEST= data set, as shown in Output 23.1.8.

```
proc print data=b;
   where _type_ = 'REDUCED' | _type_ = 'IMULT1';
run;
```

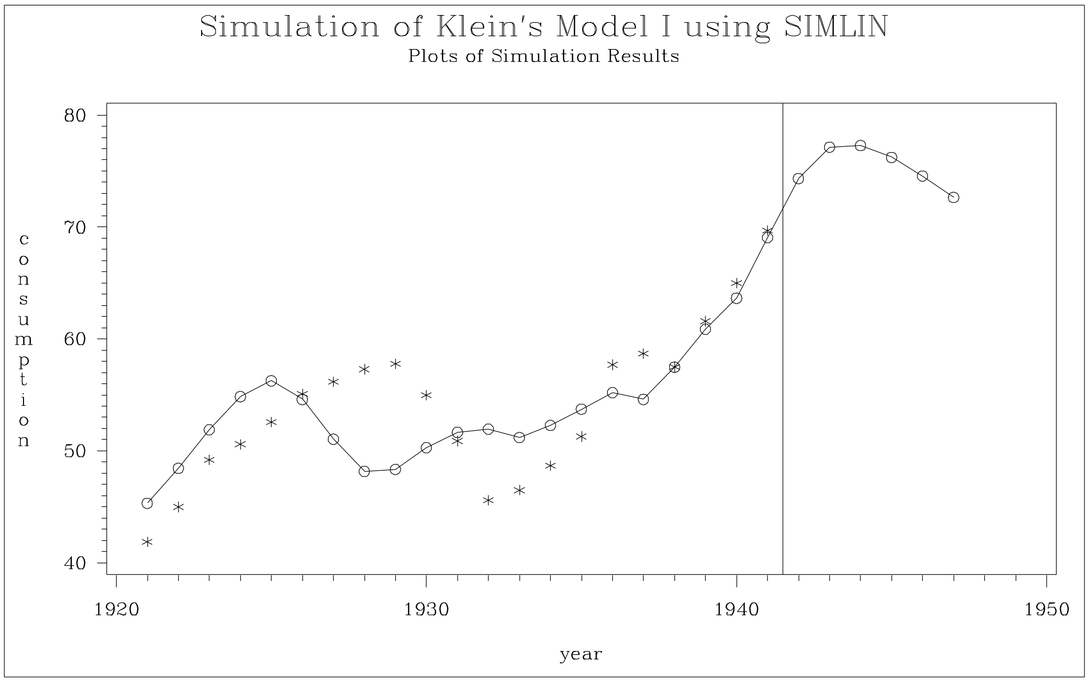
Output 23.1.8. Partial Listing of OUTEST= Data Set

```
                   Simulation of Klein's Model I using SIMLIN



           _
           D    _ _
      _    E    M S
      T    P    O I
      Y    V    D G                                          w
 O    P    A    E M                                          s
 b    E    R    L A                                          u
 s    _    _    _ _     c       p        w        i    x     m    k     y

 9 REDUCED c      .  1.63465 0.63465  1.09566  0.63465 0  0.19585 0  1.29151
10 REDUCED p      .  0.97236 0.97236 -0.34048  0.97236 0  1.10872 0  0.76825
11 REDUCED w      .  0.64957 0.64957  1.44059  0.64957 0  0.07263 0  0.51321
12 REDUCED i      . -0.01272 0.98728  0.00445 -0.01272 0 -0.01450 0 -0.01005
13 REDUCED x      .  1.62194 1.62194  1.10011  1.62194 0  0.18135 0  1.28146
14 REDUCED wsum   .  0.64957 0.64957  1.44059  0.64957 0  0.07263 0  1.51321
15 REDUCED k      . -0.01272 0.98728  0.00445 -0.01272 0 -0.01450 1 -0.01005
16 REDUCED y      .  1.62194 1.62194  1.10011  0.62194 1  0.18135 0  1.28146
17 IMULT1  c      .   .       .        .        .      .   .      .   .
18 IMULT1  p      .   .       .        .        .      .   .      .   .
19 IMULT1  w      .   .       .        .        .      .   .      .   .
20 IMULT1  i      .   .       .        .        .      .   .      .   .
21 IMULT1  x      .   .       .        .        .      .   .      .   .
22 IMULT1  wsum   .   .       .        .        .      .   .      .   .
23 IMULT1  k      .   .       .        .        .      .   .      .   .
24 IMULT1  y      .   .       .        .        .      .   .      .   .

                                                                           I
                                                                           n
                                                                           t
                                                                           e
                                                                           r
       k        p         x                                                c
 O     l        l         l                                                e
 b     a        a         a         w                             y        p
 s     g        g         g         p         g         t         r        t

 9 -0.12366  0.74631   0.19863   1.29151   0.63465  -0.19585   0.16399  46.7273
10 -0.18946  0.89347  -0.06173   0.76825   0.97236  -1.10872  -0.05096  42.7736
11 -0.12657  0.59687   0.26117   0.51321   0.64957  -0.07263   0.21562  31.5721
12 -0.19237  0.74404   0.00081  -0.01005  -0.01272   0.01450   0.00067  27.6184
13 -0.31603  1.49034   0.19944   1.28146   1.62194  -0.18135   0.16466  74.3457
14 -0.12657  0.59687   0.26117   1.51321   0.64957  -0.07263   0.21562  31.5721
15  0.80763  0.74404   0.00081  -0.01005  -0.01272   0.01450   0.00067  27.6184
16 -0.31603  1.49034   0.19944   1.28146   1.62194  -1.18135   0.16466  74.3457
17   .        .         .        0.82913   1.04942  -0.86526  -0.00541  43.2744
18   .        .         .        0.60921   0.77108  -0.98217  -0.05582  28.3955
19   .        .         .        0.79449   1.00558  -0.71096   0.01250  41.4512
20   .        .         .        0.57457   0.72723  -0.82787  -0.03791  26.5723
21   .        .         .        1.40370   1.77666  -1.69313  -0.04332  69.8467
22   .        .         .        0.79449   1.00558  -0.71096   0.01250  41.4512
23   .        .         .        0.56452   0.71451  -0.81337  -0.03725  54.1907
24   .        .         .        1.40370   1.77666  -1.69313  -0.04332  69.8467
```

The actual and predicted values for the variable C are plotted in Output 23.1.9.

```
title2 h=1 'Plots of Simulation Results';
symbol1 i=none v=star;
symbol2 i=join v=circle;
proc gplot data=c;
   plot c*year=1 chat*year=2 / overlay href=1941.5;
run;
```

Output 23.1.9. Plot of Actual and Predicted Consumption



Example 23.2. Multipliers for a Third-Order System

This example shows how to fit and simulate a single equation dynamic model with third-order lags. It then shows how to convert the third-order equation into a three equation system with only first-order lags, so that the SIMLIN procedure can compute multipliers. (See the section "Multipliers for Higher Order Lags" earlier in this chapter for more information.)

The input data set TEST is created from simulated data. A partial listing of the data set TEST produced by PROC PRINT is shown in Output 23.2.1.

```
title1 'Simulate Equation with Third-Order Lags';
title2 'Listing of Simulated Input Data';
proc print data=test(obs=10);
run;
```

Output 23.2.1. Partial Listing of Input Data Set

```
                        Simulate Equation with Third-Order Lags
                            Listing of Simulated Input Data

        Obs        y        ylag1      ylag2      ylag3        x        n

          1     8.2369     8.5191     6.9491     7.8800    -1.2593      1
          2     8.6285     8.2369     8.5191     6.9491    -1.6805      2
          3    10.2223     8.6285     8.2369     8.5191    -1.9844      3
          4    10.1372    10.2223     8.6285     8.2369    -1.7855      4
          5    10.0360    10.1372    10.2223     8.6285    -1.8092      5
          6    10.3560    10.0360    10.1372    10.2223    -1.3921      6
          7    11.4835    10.3560    10.0360    10.1372    -2.0987      7
          8    10.8508    11.4835    10.3560    10.0360    -1.8788      8
          9    11.2684    10.8508    11.4835    10.3560    -1.7154      9
         10    12.6310    11.2684    10.8508    11.4835    -1.8418     10
```

The REG procedure processes the input data and writes the parameter estimates to the OUTEST= data set A.

```
title2 'Estimated Parameters';
proc reg data=test outest=a;
   model y=ylag3 x;
run;

title2 'Listing of OUTEST= Data Set';
proc print data=a;
run;
```

Output 23.2.2 shows the printed output produced by the REG procedure, and Output 23.2.3 displays the OUTEST= data set A produced.

Output 23.2.2. Estimates and Fit Information from PROC REG

```
                    Simulate Equation with Third-Order Lags
                             Estimated Parameters

                              The REG Procedure
                                Model: MODEL1
                            Dependent Variable: y

                             Analysis of Variance

                                      Sum of           Mean
Source                    DF         Squares         Square    F Value    Pr > F

Model                      2       173.98377       86.99189    1691.98    <.0001
Error                     27         1.38818        0.05141
Corrected Total           29       175.37196


            Root MSE              0.22675    R-Square     0.9921
            Dependent Mean       13.05234    Adj R-Sq     0.9915
            Coeff Var             1.73721


                             Parameter Estimates

                            Parameter       Standard
     Variable     DF         Estimate          Error    t Value    Pr > |t|

     Intercept     1          0.14239        0.23657       0.60      0.5523
     ylag3         1          0.77121        0.01723      44.77      <.0001
     x             1         -1.77668        0.10843     -16.39      <.0001
```

Output 23.2.3. The OUTEST= Data Set Created by PROC REG

```
                    Simulate Equation with Third-Order Lags
                         Listing of OUTEST= Data Set

  Obs  _MODEL_  _TYPE_  _DEPVAR_   _RMSE_  Intercept   ylag3       x       y

   1   MODEL1   PARMS      y      0.22675   0.14239   0.77121  -1.77668   -1
```

The SIMLIN procedure processes the TEST data set using the estimates from PROC REG. The following statements perform the simulation and write the results to the OUT= data set OUT2.

```
title2 'Simulation of Equation';
proc simlin est=a data=test nored;
   endogenous y;
   exogenous  x;
   lagged ylag3 y 3;
   id n;
   output out=out1 predicted=yhat residual=yresid;
run;
```

The printed output from the SIMLIN procedure is shown in Output 23.2.4.

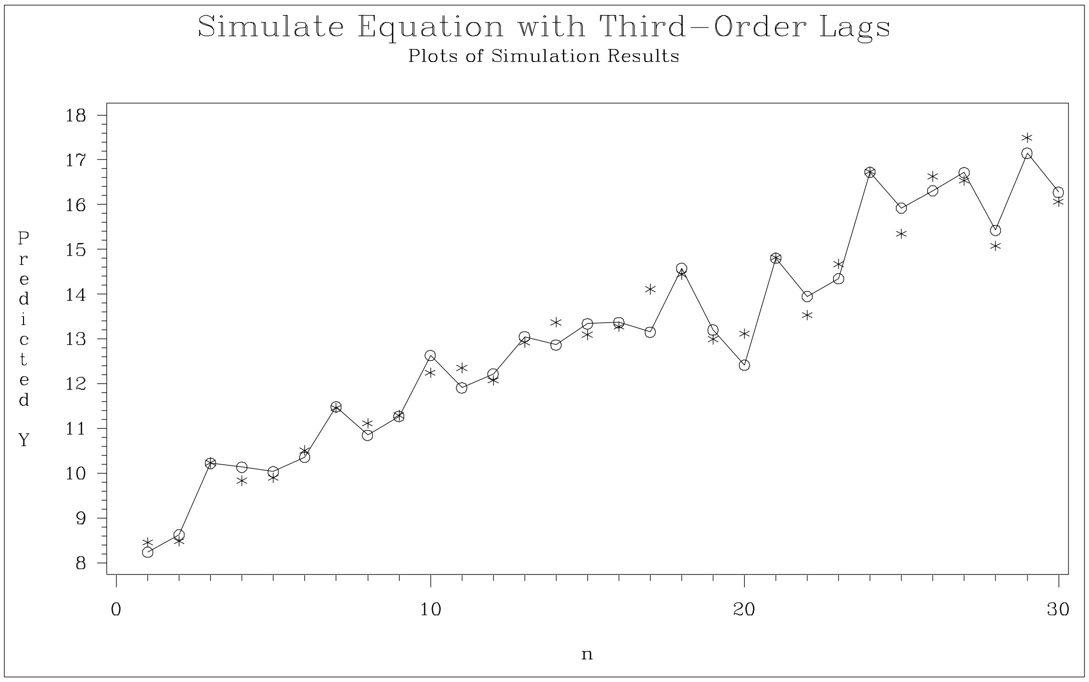
Output 23.2.4. Output Produced by PROC SIMLIN

```
                        Simulate Equation with Third-Order Lags
                                 Simulation of Equation

                                  The SIMLIN Procedure

                                    Fit Statistics

                             Mean   Mean Pct   Mean Abs    Mean Abs        RMS    RMS Pct
Variable          N         Error      Error      Error   Pct Error      Error      Error

y                30       -0.0233    -0.2268     0.2662     2.05684     0.3408     2.6159
```

The following statements plot the actual and predicted values, as shown in Output 23.2.5.

```
title2 'Plots of Simulation Results';
symbol1 i=none v=star;
symbol2 i=join v=circle;
proc gplot data=out1;
   plot yhat*n=1 y*n=2 / overlay;
run;
```

Output 23.2.5. Plot of Predicted and Actual Values



Next, the input data set TEST is modified by creating two new variables, YLAG1X and YLAG2X, that are equal to YLAG1 and YLAG2. These variables are used in the SYSLIN procedure. (The estimates produced by PROC SYSLIN are the same as before and are not shown.) A listing of the OUTEST= data set B created by PROC SYSLIN is shown in Output 23.2.6.

```
data test2;
   set test;
   ylag1x=ylag1;
   ylag2x=ylag2;
run;

title2 'Estimation of parameters and definition of identities';
proc syslin data=test2 outest=b;
   endogenous y ylag1x ylag2x;
   model y=ylag3 x;
   identity ylag1x=ylag1;
   identity ylag2x=ylag2;
run;

title2 'Listing of OUTEST= data set from PROC SYSLIN';
proc print data=b;
run;
```

Output 23.2.6. Listing of OUTEST= Data Set Created from PROC SYSLIN

```
                   Simulate Equation with Third-Order Lags
                 Listing of OUTEST= data set from PROC SYSLIN

                                           I
                _        _                 n
                S      _ D         _       t
  _             T      M E         S       e                            y  y
  T             A      O P         I       r       y            y y     l  l
  Y             T      D V         G       c       l            l l     a  a
O P             U      E A         M       e       a            a a     g  g
b E             S      L R         A       p       g            g g     1  2
s _             _      _ _         _       t       3        x   1 2  y  x  x

1 OLS      0 Converged y y      0.22675 0.14239 0.77121 -1.77668 . . -1  .  .
2 IDENTITY 0 Converged   ylag1x  .     0.00000   .        .     1 .  . -1  .
3 IDENTITY 0 Converged   ylag2x  .     0.00000   .        .     . 1  .  . -1
```

The SIMLIN procedure is used to compute the reduced form and multipliers. The OUTEST= data set B from PROC SYSLIN is used as the EST= data set for the SIMLIN procedure. The following statements perform the multiplier analysis.

```
title2 'Simulation of transformed first-order equation system';

proc simlin est=b data=test2 total interim=2;
   endogenous y ylag1x ylag2x;
   exogenous  x;
   lagged  ylag1 y 1  ylag2 ylag1x 1  ylag3 ylag2x 1;
   id n;
   output out=out2 predicted=yhat residual=yresid;
run;
```

Output 23.2.7 shows the interim 2 and total multipliers printed by the SIMLIN procedure.

Output 23.2.7. Interim 2 and Total Multipliers

```
                    Simulate Equation with Third-Order Lags
             Simulation of transformed first-order equation system

                              The SIMLIN Procedure

                       Interim Multipliers for Interim 2

                     Variable                x      Intercept

                     y                0.000000      0.0000000
                     ylag1x           0.000000      0.0000000
                     ylag2x          -1.776682      0.1423865


                               Total Multipliers

                     Variable                x      Intercept

                     y               -7.765556      0.6223455
                     ylag1x          -7.765556      0.6223455
                     ylag2x          -7.765556      0.6223455
```

References

Maddala, G.S (1977), *Econometrics*, New York: McGraw-Hill Book Co.

Pindyck, R.S. and Rubinfeld, D.L. (1991), *Econometric Models and Economic Forecasts*, Third Edition, New York: McGraw-Hill Book Co.

Theil, H. (1971), *Principles of Econometrics*, New York: John Wiley & Sons, Inc.

Chapter 24
The SPECTRA Procedure

Chapter Contents

Chapter 24
The SPECTRA Procedure

Overview

The SPECTRA procedure performs spectral and cross-spectral analysis of time series. You can use spectral analysis techniques to look for periodicities or cyclical patterns in data.

The SPECTRA procedure produces estimates of the spectral and cross-spectral densities of a multivariate time series. Estimates of the spectral and cross-spectral densities of a multivariate time series are produced using a finite Fourier transform to obtain periodograms and cross-periodograms. The periodogram ordinates are smoothed by a moving average to produce estimated spectral and cross-spectral densities. PROC SPECTRA can also test whether or not the data are white noise.

PROC SPECTRA uses the finite Fourier transform to decompose data series into a sum of sine and cosine waves of different amplitudes and wavelengths. The Fourier transform decomposition of the series x_t is

$$x_t = \frac{a_0}{2} + \sum_{k=1}^{m} [a_k cos(\omega_k t) + b_k sin(\omega_k t)]$$

where

t	is the time subscript, $t = 1, 2, \ldots , n$
x_t	are the data
n	is the number of observations in the time series
m	is the number of of frequencies in the Fourier decomposition: $m = \frac{n}{2}$ if n is even; $m = \frac{n-1}{2}$ if n is odd
a_0	is the mean term: $a_0 = 2\overline{x}$
a_k	are the cosine coefficients
b_k	are the sine coefficients
ω_k	are the Fourier frequencies: $\omega_k = \frac{2\pi k}{n}$

Functions of the Fourier coefficients a_k and b_k can be plotted against frequency or against wave length to form *periodograms*. The amplitude periodogram J_k is defined as follows:

$$J_k = \frac{n}{2}(a_k^2 + b_k^2)$$

Several definitions of the term periodogram are used in the spectral analysis literature. The following discussion refers to the J_k sequence as the periodogram.

The periodogram can be interpreted as the contribution of the *k*th harmonic ω_k to the total sum of squares, in an analysis of variance sense, for the decomposition of the process into two-degree-of-freedom components for each of the *m* frequencies. When *n* is even, $sin(\omega_{\frac{n}{2}})$ is zero, and thus the last periodogram value is a one-degree-of-freedom component.

The periodogram is a volatile and inconsistent estimator of the spectrum. The spectral density estimate is produced by smoothing the periodogram. Smoothing reduces the variance of the estimator but introduces a bias. The weight function used for the smoothing process, W(), often called the kernel or spectral window, is specified with the WEIGHTS statement. It is related to another weight function, *w*(), the lag window, that is used in other methods to taper the correlogram rather than to smooth the periodogram. Many specific weighting functions have been suggested in the literature (Fuller 1976, Jenkins and Watts 1968, Priestly 1981). Table 24.1 later in this chapter gives the formulas relevant when the WEIGHTS statement is used.

Letting *i* represent the imaginary unit $\sqrt{-1}$, the cross-periodogram is defined as follows:

$$J_k^{xy} = \frac{n}{2}(a_k^x a_k^y + b_k^x b_k^y) + i\frac{n}{2}(a_k^x b_k^y - b_k^x a_k^y)$$

The cross-spectral density estimate is produced by smoothing the cross-periodogram in the same way as the periodograms are smoothed using the spectral window specified by the WEIGHTS statement.

The SPECTRA procedure creates an output SAS data set whose variables contain values of the periodograms, cross-periodograms, estimates of spectral densities, and estimates of cross-spectral densities. The form of the output data set is described in the section "OUT= Data Set" later in this chapter.

Getting Started

To use the SPECTRA procedure, specify the input and output data sets and options for the analysis you want on the PROC SPECTRA statement, and list the variables to analyze in the VAR statement.

For example, to take the Fourier transform of a variable X in a data set A, use the following statements:

```
proc spectra data=a out=b coef;
   var x;
run;
```

This PROC SPECTRA step writes the Fourier coefficients a_k and b_k to the variables COS_01 and SIN_01 in the output data set B.

When a WEIGHTS statement is specified, the periodogram is smoothed by a weighted moving average to produce an estimate for the spectral density of the series. The following statements write a spectral density estimate for X to the variable S_01 in the output data set B.

```
proc spectra data=a out=b s;
   var x;
   weights 1 2 3 4 3 2 1;
run;
```

When the VAR statement specifies more than one variable, you can perform cross-spectral analysis by specifying the CROSS option. The CROSS option by itself produces the cross-periodograms. For example, the following statements write the real and imaginary parts of the cross-periodogram of X and Y to the variable RP_01_02 and IP_01_02 in the output data set B.

```
proc spectra data=a out=b cross;
   var x y;
run;
```

To produce cross-spectral density estimates, combine the CROSS option and the S option. The cross-periodogram is smoothed using the weights specified by the WEIGHTS statement in the same way as the spectral density. The squared coherency and phase estimates of the cross-spectrum are computed when the K and PH options are used.

The following example computes cross-spectral density estimates for the variables X and Y.

```
proc spectra data=a out=b cross s;
   var x y;
   weights 1 2 3 4 3 2 1;
run;
```

The real part and imaginary part of the cross-spectral density estimates are written to the variable CS_01_02 and QS_01_02, respectively.

Syntax

The following statements are used with the SPECTRA procedure.

> **PROC SPECTRA** *options*;
>   **BY** *variables*;
>   **VAR** *variables*;
>   **WEIGHTS** *constants*;

Functional Summary

The statements and options controlling the SPECTRA procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify the variables to be analyzed	VAR	
specify weights for spectral density estimates	WEIGHTS	
Data Set Options		
specify the input data set	PROC SPECTRA	DATA=
specify the output data set	PROC SPECTRA	OUT=
Output Control Options		
output the amplitudes of the cross-spectrum	PROC SPECTRA	A
output the Fourier coefficients	PROC SPECTRA	COEF
output the periodogram	PROC SPECTRA	P
output the spectral density estimates	PROC SPECTRA	S
output cross-spectral analysis results	PROC SPECTRA	CROSS
output squared coherency of the cross-spectrum	PROC SPECTRA	K
output the phase of the cross-spectrum	PROC SPECTRA	PH
Smoothing Options		
specify the Bartlett kernel	WEIGHTS	BART
specify the Parzen kernel	WEIGHTS	PARZEN
specify the Quadratic Spectral kernel	WEIGHTS	QS
specify the Tukey-Hanning kernel	WEIGHTS	TUKEY
specify the Truncated kernel	WEIGHTS	TRUNCAT

Description	**Statement**	**Option**
Other Options		
subtract the series mean	PROC SPECTRA	ADJMEAN
specify an alternate quadrature spectrum estimate	PROC SPECTRA	ALTW
request tests for white noise	PROC SPECTRA	WHITETEST

PROC SPECTRA Statement

PROC SPECTRA *options;*

The following options can be used in the PROC SPECTRA statement.

A

outputs the amplitude variables (A_*nn*_*mm*) of the cross-spectrum.

ADJMEAN
CENTER

subtracts the series mean before performing the Fourier decomposition. This sets the first periodogram ordinate to 0 rather than 2*n* times the squared mean. This option is commonly used when the periodograms are to be plotted to prevent a large first periodogram ordinate from distorting the scale of the plot.

ALTW

specifies that the quadrature spectrum estimate is computed at the boundaries in the same way as the spectral density estimate and the cospectrum estimate are computed.

COEF

outputs the Fourier cosine and sine coefficients of each series, in addition to the periodogram.

CROSS

is used with the P and S options to output cross-periodograms and cross-spectral densities.

DATA= *SAS-data-set*

names the SAS data set containing the input data. If the DATA= option is omitted, the most recently created SAS data set is used.

K

outputs the squared coherency variables (K_*nn*_*mm*) of the cross-spectrum. The K_*nn*_*mm* variables are identically 1 unless weights are given in the WEIGHTS statement and the S option is specified.

OUT= *SAS-data-set*
names the output data set created by PROC SPECTRA to store the results. If the OUT= option is omitted, the output data set is named using the DATA*n* convention.

P
outputs the periodogram variables. The variables are named P_*nn*, where *nn* is an index of the original variable with which the periodogram variable is associated. When both the P and CROSS options are specified, the cross-periodogram variables RP_*nn*_*mm* and IP_*nn*_*mm* are also output.

PH
outputs the phase variables (PH_*nn*_*mm*) of the cross-spectrum.

S
outputs the spectral density estimates. The variables are named S_*nn*, where *nn* is an index of the original variable with which the estimate variable is associated. When both the S and CROSS options are specified, the cross-spectral variables CS_*nn*_*mm* and QS_*nn*_*mm* are also output.

WHITETEST
prints a test of the hypothesis that the series are white noise. See "White Noise Test" later in this chapter for details.

Note that the CROSS, A, K, and PH options are only meaningful if more than one variable is listed in the VAR statement.

BY Statement

BY *variables;*

A BY statement can be used with PROC SPECTRA to obtain separate analyses for groups of observations defined by the BY variables.

VAR Statement

VAR *variables;*

The VAR statement specifies one or more numeric variables containing the time series to analyze. The order of the variables in the VAR statement list determines the index, *nn*, used to name the output variables. The VAR statement is required.

WEIGHTS Statement

WEIGHTS *constant-specification* | *kernel-specification;*

The WEIGHTS statement specifies the relative weights used in the moving average applied to the periodogram ordinates to form the spectral density estimates. A WEIGHTS statement must be used to produce smoothed spectral density estimates. If the WEIGHTS statement is not used, only the periodogram is produced.

Using Constant Specifications

Any number of weighting constants can be specified. The constants should be positive and symmetric about the middle weight. The middle constant, (or the constant to the right of the middle if an even number of weight constants are specified), is the relative weight of the current periodogram ordinate. The constant immediately following the middle one is the relative weight of the next periodogram ordinate, and so on. The actual weights used in the smoothing process are the weights specified in the WEIGHTS statement scaled so that they sum to $\frac{1}{4\pi}$.

The moving average reflects at each end of the periodogram. The first periodogram ordinate is not used; the second periodogram ordinate is used in its place.

For example, a simple triangular weighting can be specified using the following WEIGHTS statement:

```
weights 1 2 3 2 1;
```

Using Kernel Specifications

You can specify five different kernels in the WEIGHTS statement. The syntax for the statement is

WEIGHTS [PARZEN][BART][TUKEY][TRUNCAT][QS] [c e];

where $c >= 0$ and $e >= 0$ are used to compute the bandwidth parameter as

$$\mathrm{l(q) = cq^e}$$

and q is the number of periodogram ordinates +1:

$$q = \mathrm{floor(n/2) + 1}$$

To specify the bandwidth explicitly, set $c =$ to the desired bandwidth and $e = 0$.

For example, a Parzen kernel can be specified using the following WEIGHTS statement:

```
weights parzen 0.5 0;
```

For details, see the “Kernels” section on page 1400, later in this chapter.

Details

Input Data

Observations in the data set analyzed by the SPECTRA procedure should form ordered, equally spaced time series. No more than 99 variables can be included in the analysis.

Data are often de-trended before analysis by the SPECTRA procedure. This can be done by using the residuals output by a SAS regression procedure. Optionally, the data can be centered using the ADJMEAN option in the PROC SPECTRA statement, since the zero periodogram ordinate corresponding to the mean is of little interest from the point of view of spectral analysis.

Missing Values

Missing values are essentially excluded from the analysis by the SPECTRA procedure. If the SPECTRA procedure encounters missing values for any variable listed in the VAR statement, the procedure determines the longest contiguous span of data that has no missing values for the variables listed in the VAR statement and uses it for the analysis.

Computational Method

If the number of observations *n* factors into prime integers that are less than or equal to 23, and the product of the square-free factors of *n* is less than 210, then PROC SPECTRA uses the Fast Fourier Transform developed by Cooley and Tukey and implemented by Singleton (1969). If *n* cannot be factored in this way, then PROC SPECTRA uses a Chirp-Z algorithm similar to that proposed by Monro and Branch (1976). To reduce memory requirements, when *n* is small the Fourier coefficients are computed directly using the defining formulas.

Kernels

Kernels are used to smooth the periodogram by using a weighted moving average of nearby points. A smoothed periodogram is defined by the following equation.

$$\hat{J}_i(\mathrm{l(q)}) = \sum_{\tau=-\mathrm{l(q)}}^{\mathrm{l(q)}} \mathrm{w}\left(\frac{\tau}{\mathrm{l(q)}}\right) \tilde{\mathrm{J}}_{\mathrm{i}+\tau}$$

where $\mathrm{w(x)}$ is the kernel or weight function. At the endpoints, the moving average is computed cyclically; that is,

$$\tilde{J}_{i+\tau} = \begin{cases} J_{i+\tau} & 0 <= \mathrm{i}+\tau <= \mathrm{q} \\ J_{-(i+\tau)} & \mathrm{i}+\tau < 0 \\ J_{q-(i+\tau)} & \mathrm{i}+\tau > \mathrm{q} \end{cases}$$

The SPECTRA procedure supports the following kernels. They are listed with their default bandwidth functions.

Bartlett: KERNEL BART

$$\begin{aligned} \mathrm{w(x)} &= \begin{cases} 1-|x| & |x| \le 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l(q)} &= \frac{1}{2} q^{1/3} \end{aligned}$$

Parzen: KERNEL PARZEN

$$\begin{aligned} \mathrm{w(x)} &= \begin{cases} 1-6|x|^2+6|x|^3 & 0 \le |x| \le \frac{1}{2} \\ 2(1-|x|)^3 & \frac{1}{2} \le |x| \le 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l(q)} &= q^{1/5} \end{aligned}$$

Quadratic Spectral: KERNEL QS

$$\begin{aligned} \mathrm{w(x)} &= \frac{25}{12\pi^2 x^2} \left(\frac{sin(6\pi x/5)}{6\pi x/5} - cos(6\pi x/5) \right) \\ \mathrm{l(q)} &= \frac{1}{2} q^{1/5} \end{aligned}$$

Tukey-Hanning: KERNEL TUKEY

$$\begin{aligned} \mathrm{w(x)} &= \begin{cases} (1+cos(\pi x))/2 & |x| \le 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l(q)} &= \frac{2}{3} q^{1/5} \end{aligned}$$

Truncated: KERNEL TRUNCAT

$$\begin{aligned} \mathrm{w(x)} &= \begin{cases} 1 & |x| \le 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l(q)} &= \frac{1}{4} q^{1/5} \end{aligned}$$

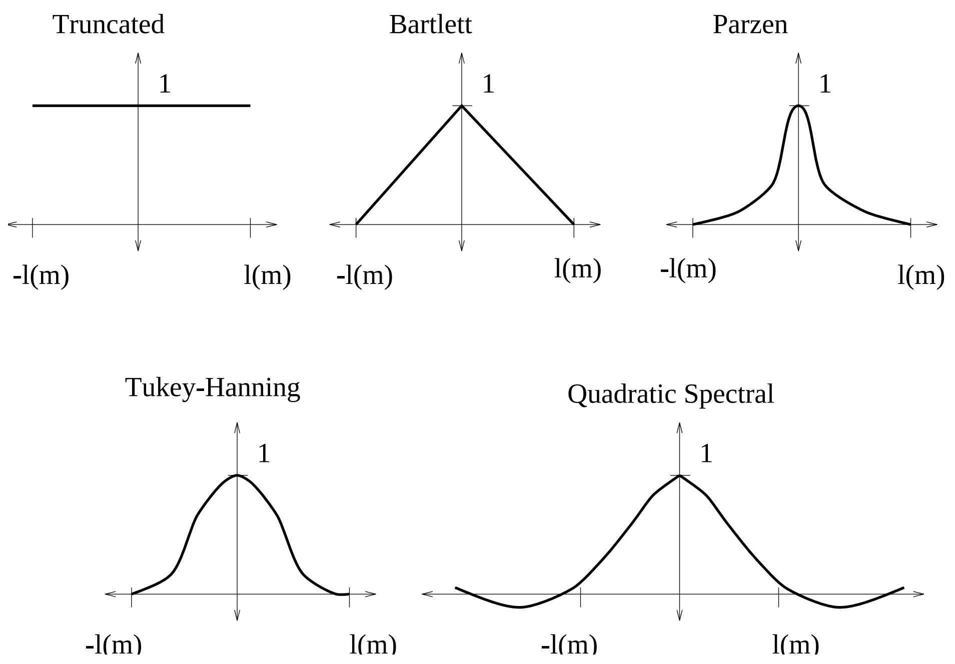


Figure 24.1. Kernels for Smoothing

Refer to Andrews (1991) for details on the properties of these kernels.

White Noise Test

PROC SPECTRA prints two test statistics for white noise when the WHITETEST option is specified: Fisher's Kappa (Davis 1941, Fuller 1976) and Bartlett's Kolmogorov-Smirnov statistic (Bartlett 1966, Fuller 1976, Durbin 1967).

If the time series is a sequence of independent random variables with mean 0 and variance σ^2, then the periodogram, J_k, will have the same expected value for all k. For a time series with nonzero autocorrelation, each ordinate of the periodogram, J_k, will have different expected values. The Fisher's Kappa statistic tests whether the largest J_k can be considered different from the mean of the J_k. Critical values for the Fisher's Kappa test can be found in Fuller 1976 and *SAS/ETS Software: Applications Guide 1*.

The Kolmogorov-Smirnov statistic reported by PROC SPECTRA has the same asymptotic distribution as Bartlett's test (Durbin 1967). The Kolmogorov-Smirnov statistic compares the normalized cumulative periodogram with the cumulative distribution function of a uniform(0,1) random variable. The normalized cumulative periodogram, F_j, of the series is

$$F_j = \frac{\sum_{k=1}^{j} J_k}{\sum_{k=1}^{m} J_k}, j = 1, 2 \ldots, m-1$$

where $m = \frac{n}{2}$ if n is even or $m = \frac{n-1}{2}$ if n is odd. The test statistic is the maximum absolute difference of the normalized cumulative periodogram and the uniform cumulative distribution function. For $m-1$ greater than 100, if Bartlett's Kolmogorov-Smirnov statistic exceeds the critical value

$$\frac{a}{\sqrt{m-1}}$$

where $a = 1.36$ or $a = 1.63$ corresponding to 5% or 1% significance levels respectively, then reject the null hypothesis that the series represents white noise. Critical values for $m-1 < 100$ can be found in a table of significance points of the Kolmogorov-Smirnov statistics with sample size $m-1$ (Miller 1956, Owen 1962).

Transforming Frequencies

The variable FREQ in the data set created by the SPECTRA procedure ranges from 0 to π. Sometimes it is preferable to express frequencies in cycles per observation period, which is equal to $\frac{2}{\pi}$FREQ.

To express frequencies in cycles per unit time (for example, in cycles per year), multiply FREQ by $\frac{d}{2\pi}$, where *d* is the number of observations per unit of time. For example, for monthly data, if the desired time unit is years then *d* is 12. The period of the cycle is $\frac{2\pi}{d \times \mathrm{FREQ}}$, which ranges from $\frac{2}{d}$ to infinity.

OUT= Data Set

The OUT= data set contains $\frac{n}{2}+1$ observations, if *n* is even, or $\frac{n+1}{2}$ observations, if *n* is odd, where *n* is the number of observations in the time series.

The variables in the new data set are named according to the following conventions. Each variable to be analyzed is associated with an index. The first variable listed in the VAR statement is indexed as 01, the second variable as 02, and so on. Output variables are named by combining indexes with prefixes. The prefix always identifies the nature of the new variable, and the indices identify the original variables from which the statistics were obtained.

Variables containing spectral analysis results have names consisting of a prefix, an underscore, and the index of the variable analyzed. For example, the variable S_01 contains spectral density estimates for the first variable in the VAR statement. Variables containing cross-spectral analysis results have names consisting of a prefix, an underscore, the index of the first variable, another underscore, and the index of the second variable. For example, the variable A_01_02 contains the amplitude of the cross-spectral density estimate for the first and second variables in the VAR statement.

Table 24.1 shows the formulas and naming conventions used for the variables in the OUT= data set. Let X be variable number *nn* in the VAR statement list and let Y be variable number *mm* in the VAR statement list. Table 24.1 shows the output variables containing the results of the spectral and cross-spectral analysis of X and Y.

In Table 24.1 the following notation is used. Let W_j be the vector of $2p+1$ smoothing weights given by the WEIGHTS statement, normalized to sum to $\frac{1}{4\pi}$. The subscript of W_j runs from W_{-p} to W_p, so that W_0 is the middle weight in the WEIGHTS statement list. Let $\omega_k = \frac{2\pi k}{n}$, where $k = 0, 1, \ldots, \mathrm{floor}(\frac{n}{2})$.

Table 24.1. Variables Created by PROC SPECTRA

Variable	**Description**
FREQ	frequency in radians from 0 to π (Note: Cycles per observation is $\frac{\mathrm{FREQ}}{2\pi}$.)
PERIOD	period or wavelength: $\frac{2\pi}{\mathrm{FREQ}}$ (Note: PERIOD is missing for FREQ=0.)
COS_X COS_WAVE	cosine transform of X: $a_k^x = \frac{2}{n}\sum_{t=1}^{n} X_t \cos(\omega_k(t-1))$
SIN_X SIN_WAVE	sine transform of X: $b_k^x = \frac{2}{n}\sum_{t=1}^{n} X_t \sin(\omega_k(t-1))$
P_*nn*	periodogram of X: $J_k^x = \frac{n}{2}[(a_k^x)^2 + (b_k^x)^2]$
S_*nn*	spectral density estimate of X: $F_k^x = \sum_{j=-p}^{p} W_j J_{k+j}^x$ (except across endpoints)
RP_*nn*_*mm*	real part of cross-periodogram X and Y: $\mathrm{real}(\mathrm{J_k^{xy}}) = \frac{\mathrm{n}}{2}(\mathrm{a_k^x a_k^y + b_k^x b_k^y})$
IP_*nn*_*mm*	imaginary part of cross-periodogram of X and Y: $\mathrm{imag}(\mathrm{J_k^{xy}}) = \frac{\mathrm{n}}{2}(\mathrm{a_k^x b_k^y - b_k^x a_k^y})$
CS_*nn*_*mm*	cospectrum estimate (real part of cross-spectrum) of X and Y: $C_k^{xy} = \sum_{j=-p}^{p} W_j \mathrm{real}(\mathrm{J_{k+j}^{xy}})$ (except across endpoints)
QS_*nn*_*mm*	quadrature spectrum estimate (imaginary part of cross-spectrum) of X and Y: $Q_k^{xy} = \sum_{j=-p}^{p} W_j \mathrm{imag}(\mathrm{J_{k+j}^{xy}})$ (except across endpoints)
A_*nn*_*mm*	amplitude (modulus) of cross-spectrum of X and Y: $A_k^{xy} = \sqrt{(C_k^{xy})^2 + (Q_k^{xy})^2}$
K_*nn*_*mm*	coherency squared of X and Y: $K_k^{xy} = (A_k^{xy})^2/(F_k^x F_k^y)$
PH_*nn*_*mm*	phase spectrum in radians of X and Y: $\Phi_k^{xy} = \arctan(Q_k^{xy}/C_k^{xy})$

Printed Output

By default PROC SPECTRA produced no printed output.

When the WHITETEST option is specified, the SPECTRA procedure prints the following statistics for each variable in the VAR statement:

1. the name of the variable
2. M-1, the number of two-degree-of-freedom periodogram ordinates used in the tests
3. MAX(P(*)), the maximum periodogram ordinate
4. SUM(P(*)), the sum of the periodogram ordinates
5. Fisher's Kappa statistic
6. Bartlett's Kolmogorov-Smirnov test statistic

See "White Noise Test" earlier in this chapter for details.

ODS Table Names

PROC SPECTRA assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 24.2. ODS Tables Produced in PROC SPECTRA

ODS Table Name	**Description**	**Option**
WhiteNoiseTest	White Noise Test	WHITETEST
Kappa	Fishers Kappa	WHITETEST
Bartlett	Bartletts Kolmogorov-Smirnov Statistic	WHITETEST

Examples

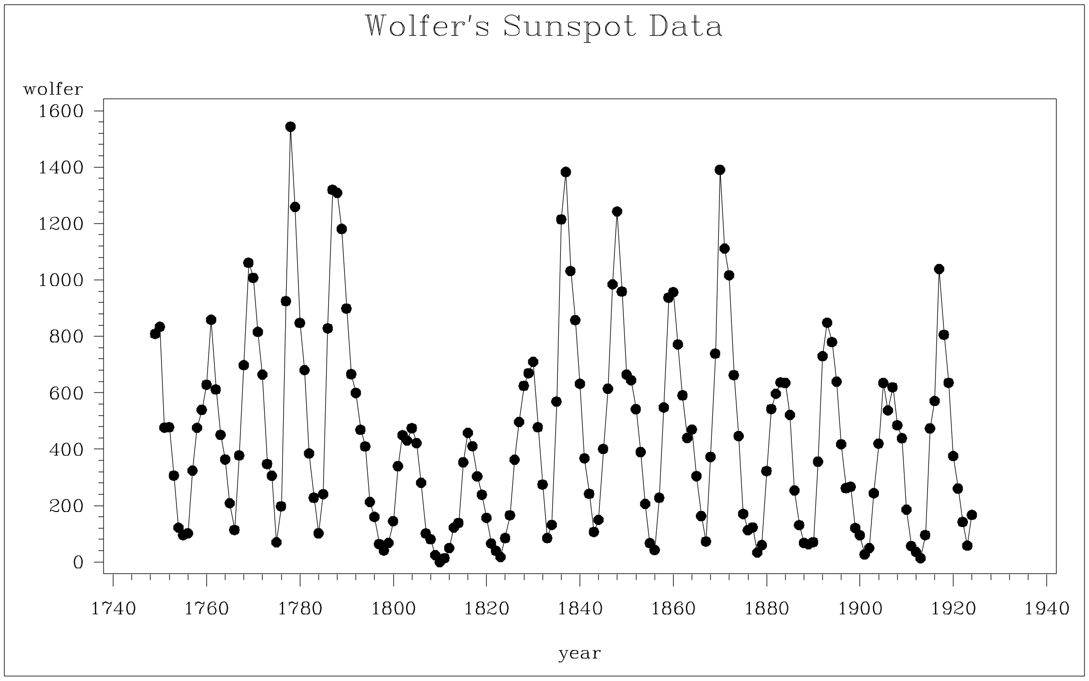
Example 24.1. Spectral Analysis of Sunspot Activity

This example analyzes Wolfer's sunspot data (Anderson 1971). The following statements read and plot the data.

```
title "Wolfer's Sunspot Data";
data sunspot;
   input year wolfer @@;
   datalines;
1749  809 1750  834 1751  477 1752  478 1753  307 1754  122 1755   96
1756  102 1757  324 1758  476 1759  540 1760  629 1761  859 1762  612
1763  451 1764  364 1765  209 1766  114 1767  378 1768  698 1769 1061
1770 1008 1771  816 1772  665 1773  348 1774  306 1775   70 1776  198
1777  925 1778 1544 1779 1259 1780  848 1781  681 1782  385 1783  228
1784  102 1785  241 1786  829 1787 1320 1788 1309 1789 1181 1790  899
1791  666 1792  600 1793  469 1794  410 1795  213 1796  160 1797   64
1798   41 1799   68 1800  145 1801  340 1802  450 1803  431 1804  475
1805  422 1806  281 1807  101 1808   81 1809   25 1810    0 1811   14
1812   50 1813  122 1814  139 1815  354 1816  458 1817  411 1818  304
1819  239 1820  157 1821   66 1822   40 1823   18 1824   85 1825  166
1826  363 1827  497 1828  625 1829  670 1830  710 1831  478 1832  275
1833   85 1834  132 1835  569 1836 1215 1837 1383 1838 1032 1839  858
1840  632 1841  368 1842  242 1843  107 1844  150 1845  401 1846  615
1847  985 1848 1243 1849  959 1850  665 1851  645 1852  542 1853  390
1854  206 1855   67 1856   43 1857  228 1858  548 1859  938 1860  957
1861  772 1862  591 1863  440 1864  470 1865  305 1866  163 1867   73
1868  373 1869  739 1870 1391 1871 1112 1872 1017 1873  663 1874  447
1875  171 1876  113 1877  123 1878   34 1879   60 1880  323 1881  543
1882  597 1883  637 1884  635 1885  522 1886  254 1887  131 1888   68
1889   63 1890   71 1891  356 1892  730 1893  849 1894  780 1895  640
1896  418 1897  262 1898  267 1899  121 1900   95 1901   27 1902   50
1903  244 1904  420 1905  635 1906  538 1907  620 1908  485 1909  439
1910  186 1911   57 1912   36 1913   14 1914   96 1915  474 1916  571
1917 1039 1918  806 1919  636 1920  376 1921  261 1922  142 1923   58
1924  167
;

symbol1 i=splines v=dot;
proc gplot data=sunspot;
   plot wolfer*year;
run;
```

The plot of the sunspot series is shown in Output 24.1.1.

Output 24.1.1. Plot of Original Data



The spectral analysis of the sunspot series is performed by the following statements:

```
proc spectra data=sunspot out=b p s adjmean whitetest;
   var wolfer;
   weights 1 2 3 4 3 2 1;
run;

proc print data=b(obs=12);
run;
```

The PROC SPECTRA statement specifies the P and S options to write the periodogram and spectral density estimates to the OUT= data set B. The WEIGHTS statement specifies a triangular spectral window for smoothing the periodogram to produce the spectral density estimate. The ADJMEAN option zeros the frequency 0 value and avoids the need to exclude that observation from the plots. The WHITETEST option prints tests for white noise.

The Fisher's Kappa test statistic of 16.070 is larger than the 5% critical value of 7.2, so the null hypothesis that the sunspot series is white noise is rejected.

The Bartlett's Kolmogorov-Smirnov statistic of 0.6501 is greater than

$$a\sqrt{1/(m-1)} = 1.36\sqrt{1/87} = 0.1458$$

so reject the null hypothesis that the spectrum represents white noise.

The printed output produced by PROC SPECTRA is shown in Output 24.1.2. The output data set B created by PROC SPECTRA is shown in part in Output 24.1.3.

Output 24.1.2. White Noise Test Results

```
                          Wolfer’s Sunspot Data

                            SPECTRA Procedure

                  Test for White Noise for Variable wolfer

                           M-1                 87
                           Max(P(*))      4062267
                           Sum(P(*))     21156512


                 Fisher’s Kappa: (M-1)*Max(P(*))/Sum(P(*))

                             Kappa    16.70489


                 Bartlett’s Kolmogorov-Smirnov Statistic:
              Maximum absolute difference of the standardized
              partial sums of the periodogram and the CDF of a
                       uniform(0,1) random variable.

             Test Statistic                          0.650055
```

Output 24.1.3. First 12 Observations of the OUT= Data Set

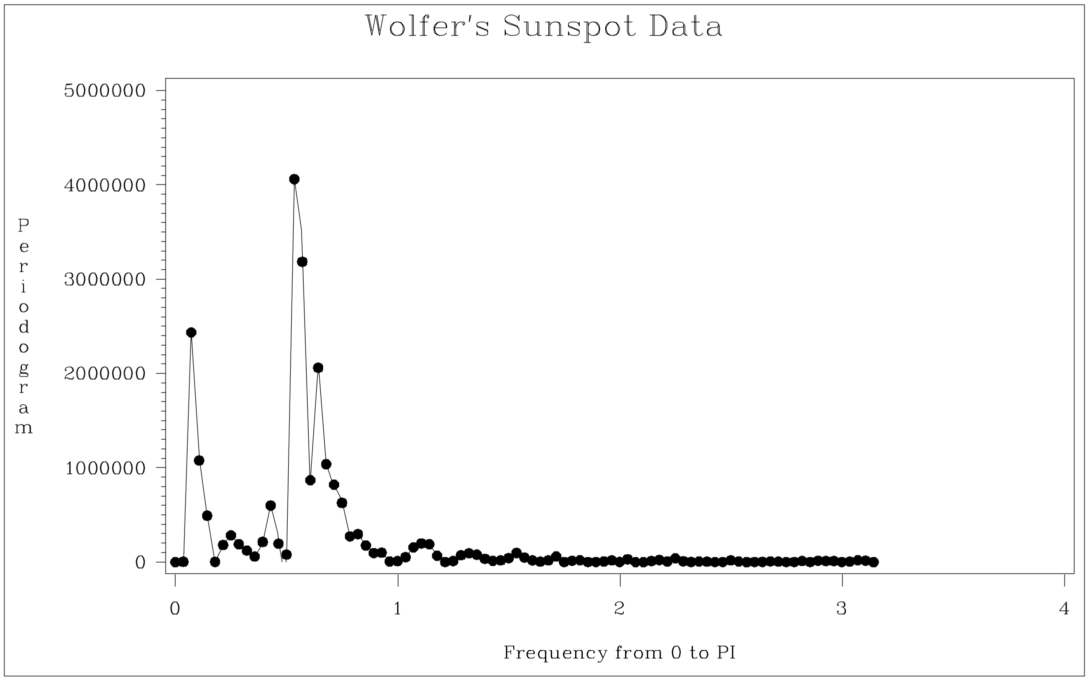
```
                          Wolfer’s Sunspot Data

              Obs       FREQ       PERIOD           P_01       S_01

                1    0.00000         .              0.00    59327.52
                2    0.03570     176.000         3178.15    61757.98
                3    0.07140      88.000      2435433.22    69528.68
                4    0.10710      58.667      1077495.76    66087.57
                5    0.14280      44.000       491850.36    53352.02
                6    0.17850      35.200         2581.12    36678.14
                7    0.21420      29.333       181163.15    20604.52
                8    0.24990      25.143       283057.60    15132.81
                9    0.28560      22.000       188672.97    13265.89
               10    0.32130      19.556       122673.94    14953.32
               11    0.35700      17.600        58532.93    16402.84
               12    0.39270      16.000       213405.16    18562.13
```

The following statements plot the periodogram and spectral density estimate:

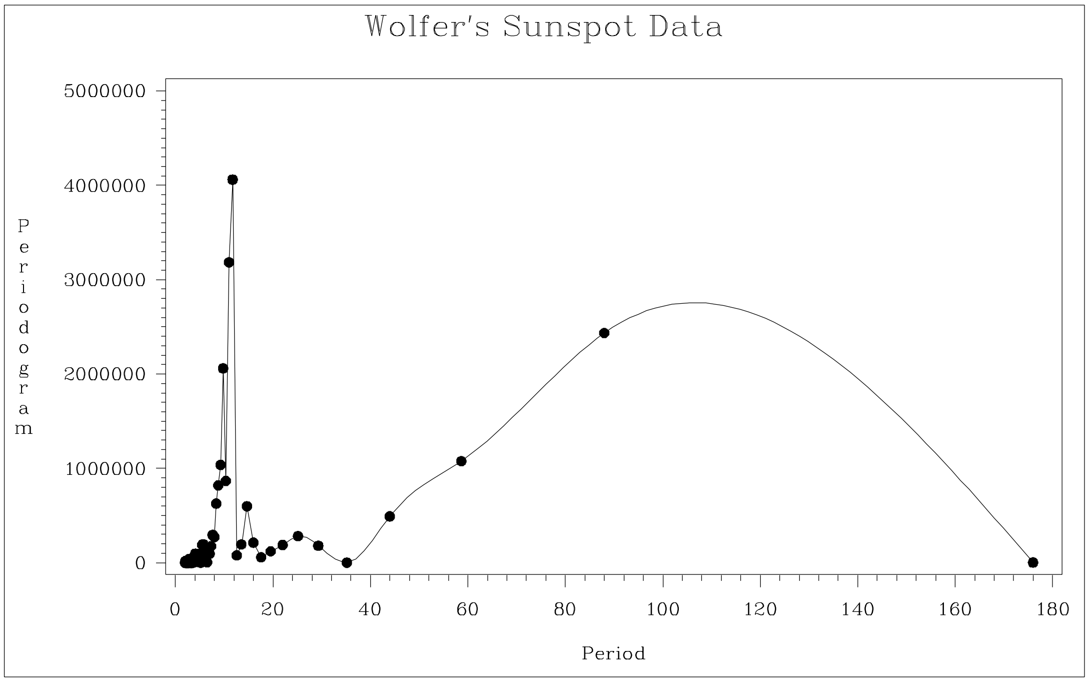
```
proc gplot data=b;
   plot p_01 * freq;
   plot p_01 * period;
   plot s_01 * freq;
   plot s_01 * period;
run;
```

The periodogram is plotted against frequency in Output 24.1.4 and plotted against period in Output 24.1.5. The spectral density estimate is plotted against frequency in Output 24.1.6 and plotted against period in Output 24.1.7.

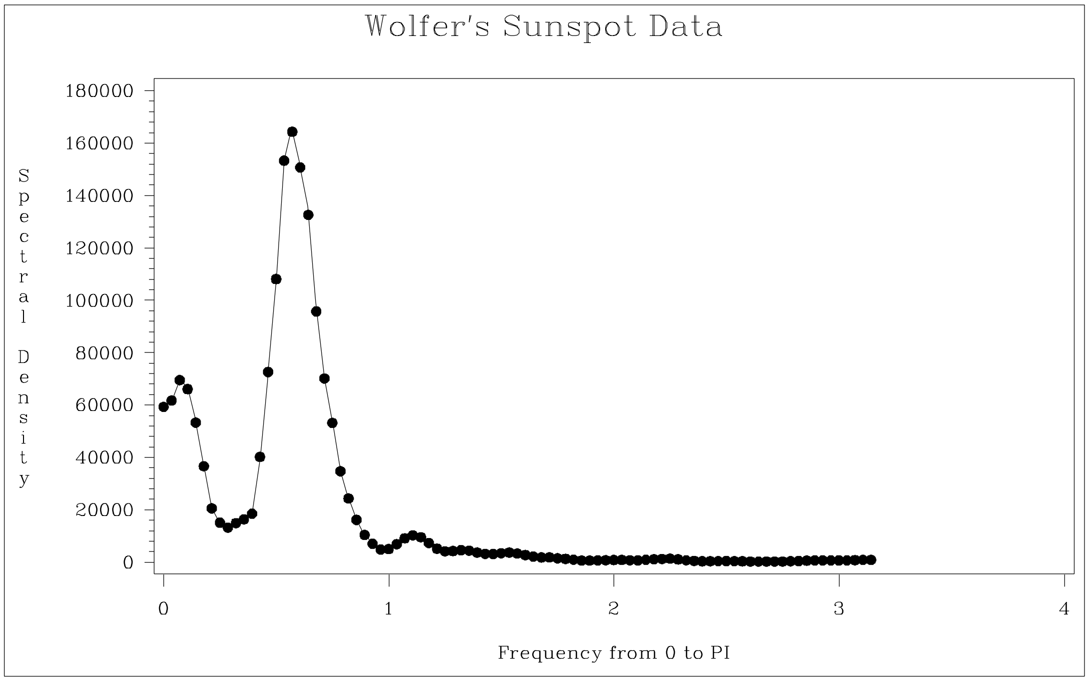
Output 24.1.4. Plot of Periodogram by Frequency



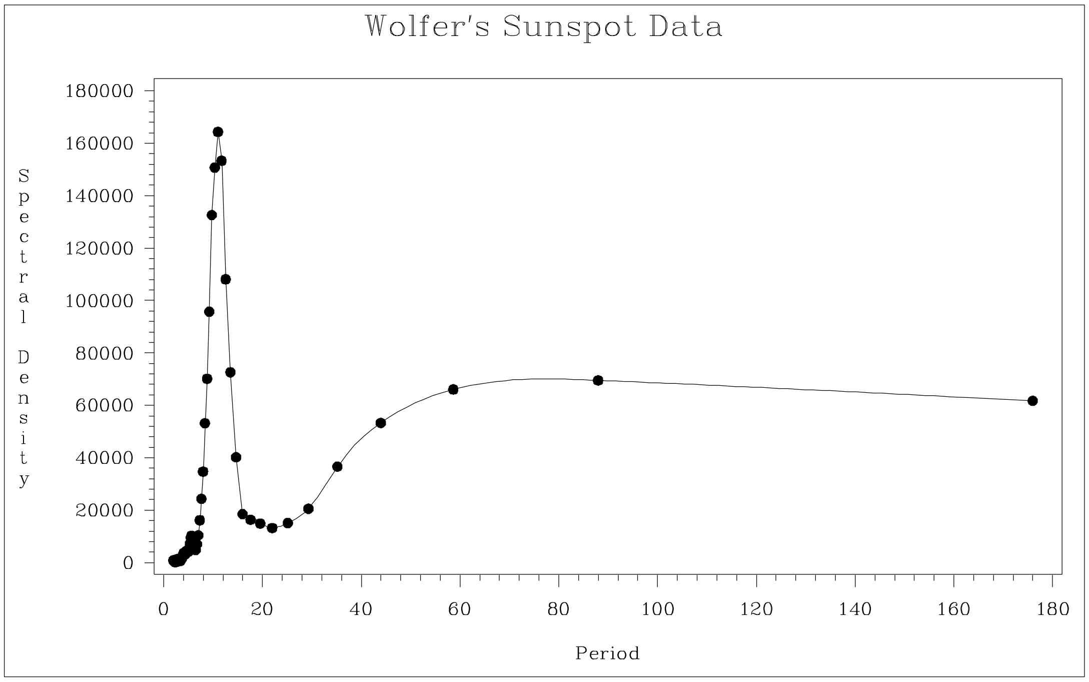
Output 24.1.5. Plot of Periodogram by Period



Output 24.1.6. Plot of Spectral Density Estimate by Frequency



Output 24.1.7. Plot of Spectral Density Estimate by Period



Since PERIOD is the reciprocal of frequency, the plot axis for PERIOD is stretched for low frequencies and compressed at high frequencies. One way to correct for this is to use a WHERE statement to restrict the plots and exclude the low frequency components. The following statements plot the spectral density for periods less than 50.

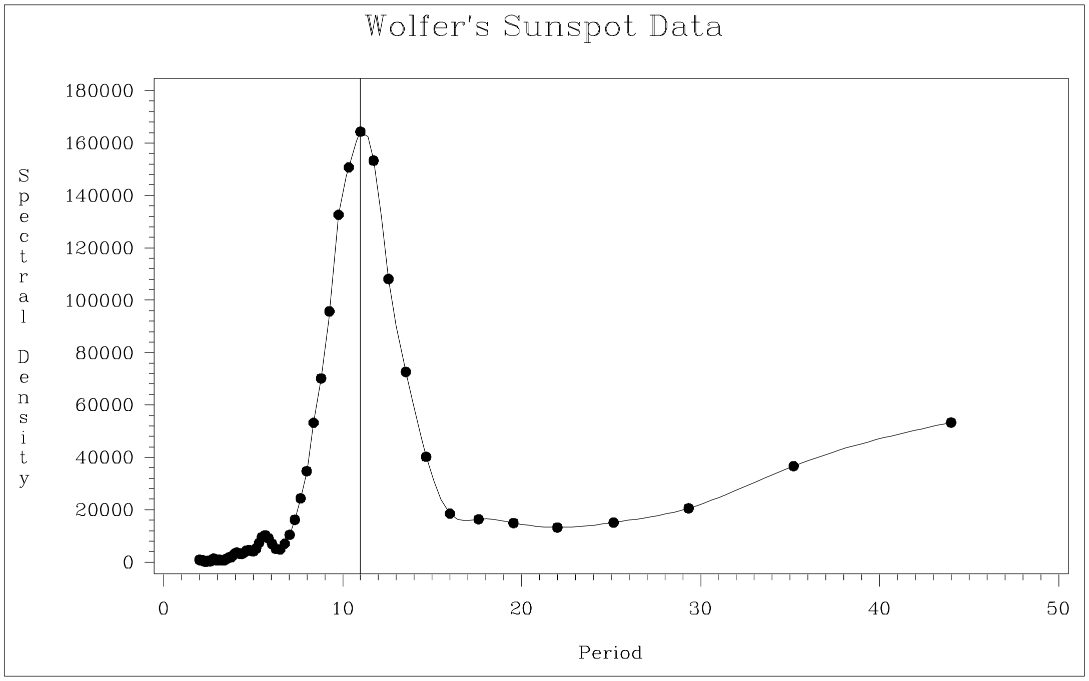
```
proc gplot data=b;
   where period < 50;
```

```
      plot s_01 * period / href=11;
   run;
```

The spectral analysis of the sunspot series confirms a strong 11-year cycle of sunspot activity. The plot makes this clear by drawing a reference line at the 11 year period, which highlights the position of the main peak in the spectral density.

Output 24.1.8 shows the plot. Contrast Output 24.1.8 with Output 24.1.7.

Output 24.1.8. Plot of Spectral Density Estimate by Period to 50 Years



Example 24.2. Cross-Spectral Analysis

This example shows cross-spectral analysis for two variables X and Y using simulated data. X is generated by an AR(1) process; Y is generated as white noise plus an input from X lagged 2 periods. All output options are specified on the PROC SPECTRA statement. PROC CONTENTS shows the contents of the OUT= data set.

```
   data a;
      xl = 0; xll = 0;
      do i = - 10 to 100;
         x = .4 * xl  + rannor(123);
         y = .5 * xll + rannor(123);
         if i > 0 then output;
         xll = xl; xl = x;
         end;
   run;

   proc spectra data=a out=b cross coef a k p ph s;
      var x y;
      weights 1 1.5 2 4 8 9 8 4 2 1.5 1;
   run;

   proc contents data=b position;
   run;
```

The PROC CONTENTS report for the output data set B is shown in Output 24.2.1.

Output 24.2.1. Contents of PROC SPECTRA OUT= Data Set

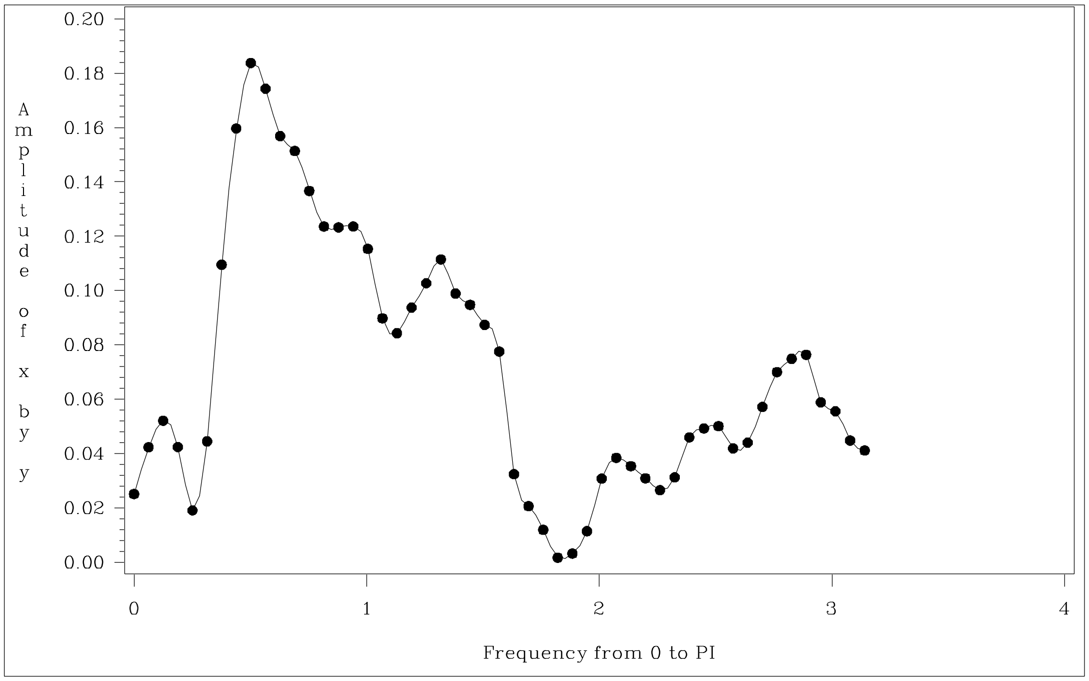
```
                              The CONTENTS Procedure

Data Set Name: WORK.B                                    Observations:         51
Member Type:   DATA                                      Variables:            17
Engine:        V8                                        Indexes:              0
Created:       12:39 Wednesday, April 28, 1999           Observation Length:   136
Last Modified: 12:39 Wednesday, April 28, 1999           Deleted Observations: 0
Protection:                                              Compressed:           NO
Data Set Type: DATA                                      Sorted:               NO
Label:         Spectral Density Estimates


                     -----Variables Ordered by Position-----

        #    Variable    Type    Len    Pos    Label
       -----------------------------------------------------------------
        1    FREQ        Num       8      0    Frequency from 0 to PI
        2    PERIOD      Num       8      8    Period
        3    COS_01      Num       8     16    Cosine Transform of x
        4    SIN_01      Num       8     24    Sine Transform of x
        5    COS_02      Num       8     32    Cosine Transform of y
        6    SIN_02      Num       8     40    Sine Transform of y
        7    P_01        Num       8     48    Periodogram of x
        8    P_02        Num       8     56    Periodogram of y
        9    S_01        Num       8     64    Spectral Density of x
       10    S_02        Num       8     72    Spectral Density of y
       11    RP_01_02    Num       8     80    Real Periodogram of x by y
       12    IP_01_02    Num       8     88    Imag Periodogram of x by y
       13    CS_01_02    Num       8     96    Cospectra of x by y
       14    QS_01_02    Num       8    104    Quadrature of x by y
       15    K_01_02     Num       8    112    Coherency**2 of x by y
       16    A_01_02     Num       8    120    Amplitude of x by y
       17    PH_01_02    Num       8    128    Phase of x by y
```

The following statements plot the amplitude of the cross-spectrum estimate against frequency and against period for periods less than 25.

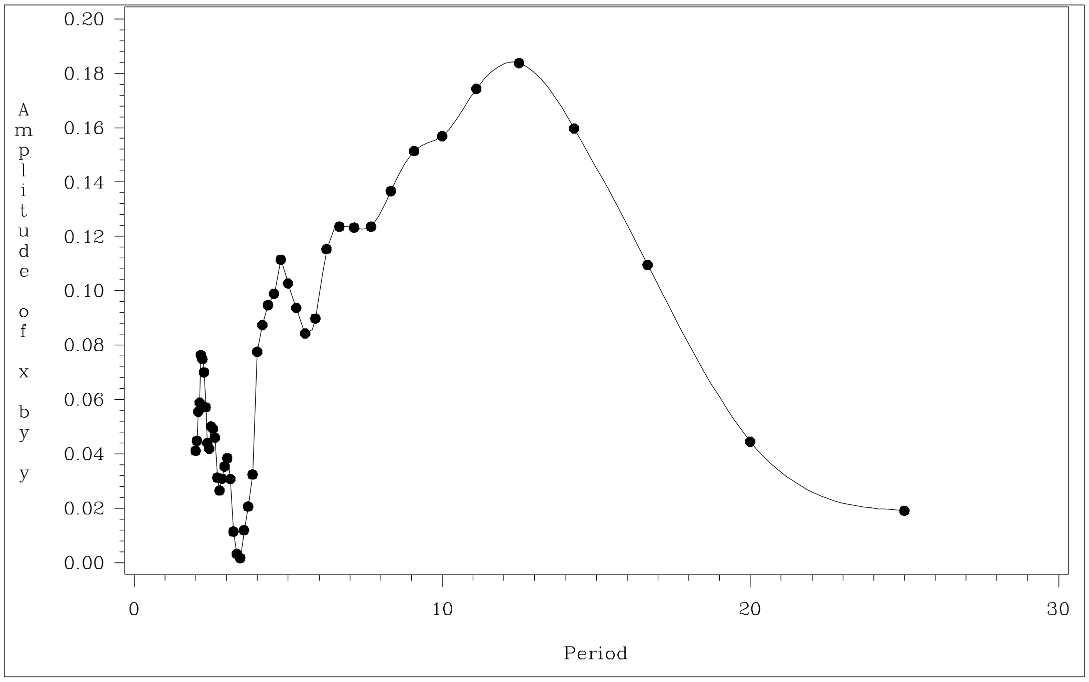
```
   symbol1 i=splines v=dot;
   proc gplot data=b;
      plot a_01_02 * freq;
   run;

   proc gplot data=b;
      plot a_01_02 * period;
      where period < 25;
   run;
```

The plot of the amplitude of the cross-spectrum estimate against frequency is shown in Output 24.2.2. The plot of the cross-spectrum amplitude against period for periods less than 25 observations is shown in Output 24.2.3.

Output 24.2.2. Plot of Cross-Spectrum Amplitude by Frequency



Output 24.2.3. Plot of Cross-Spectrum Amplitude by Period



References

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, New York: John Wiley & Sons, Inc.

Andrews, D.W.K. (1991), "Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation," *Econometrica*, 59 (3), 817-858.

Bartlett, M.S. (1966), *An Introduction to Stochastic Processes*, Second Edition, Cambridge: Cambridge University Press.

Brillinger, D.R. (1975), *Time Series: Data Analysis and Theory*, New York: Holt, Rinehart and Winston, Inc.

Davis, H.T. (1941), *The Analysis of Economic Time Series*, Bloomington, IN: Principia Press.

Durbin, J. (1967), "Tests of Serial Independence Based on the Cumulated Periodogram," *Bulletin of Int. Stat. Inst.*, 42, 1039–1049.

Fuller, W.A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons, Inc.

Gentleman, W.M. and Sande, G. (1966), "Fast Fourier transforms–for fun and profit," *AFIPS Proceedings of the Fall Joint Computer Conference*, 19, 563–578.

Jenkins, G.M. and Watts, D.G. (1968), *Spectral Analysis and Its Applications*, San Francisco: Holden-Day.

Miller, L. H. (1956), "Tables of Percentage Points of Kolmogorov Statistics," *Journal of American Statistic Association*, 51, 111.

Monro, D.M. and Branch, J.L. (1976), "Algorithm AS 117. The chirp discrete Fourier transform of general length," *Applied Statistics*, 26, 351–361.

Nussbaumer, H.J. (1982), *Fast Fourier Transform and Convolution Algorithms*, Second Edition, New York: Springer-Verlag.

Owen, D. B. (1962), *Handbook of Statistical Tables*, Addison Wesley.

Parzen, E. (1957), "On Consistent Estimates of the Spectrum of a Stationary Time Series," *Annals of Mathematical Statistics*, 28, 329-348.

Priestly, M.B. (1981), *Spectral Analysis and Time Series*, New York: Academic Press, Inc.

Singleton, R.C. (1969), "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," *I.E.E.E. Transactions of Audio and Electroacoustics*, AU-17, 93–103.

Chapter 25
The STATESPACE Procedure

Chapter Contents

Chapter 25
The STATESPACE Procedure

Overview

The STATESPACE procedure analyzes and forecasts multivariate time series using the state space model. The STATESPACE procedure is appropriate for jointly forecasting several related time series that have dynamic interactions. By taking into account the autocorrelations among the whole set of variables, the STATESPACE procedure may give better forecasts than methods that model each series separately.

By default, the STATESPACE procedure automatically selects a state space model appropriate for the time series, making the procedure a good tool for automatic forecasting of multivariate time series. Alternatively, you can specify the state space model by giving the form of the state vector and the state transition and innovation matrices.

The methods used by the STATESPACE procedure assume that the time series are jointly stationary. Nonstationary series must be made stationary by some preliminary transformation, usually by differencing. The STATESPACE procedure allows you to specify differencing of the input data. When differencing is specified, the STATESPACE procedure automatically integrates forecasts of the differenced series to produce forecasts of the original series.

The State Space Model

The *state space model* represents a multivariate time series through auxiliary variables, some of which may not be directly observable. These auxiliary variables are called the *state vector*. The state vector summarizes all the information from the present and past values of the time series relevant to the prediction of future values of the series. The observed time series are expressed as linear combinations of the state variables. The state space model is also called a Markovian representation, or a canonical representation, of a multivariate time series process. The state space approach to modeling a multivariate stationary time series is summarized in Akaike (1976).

The state space form encompasses a very rich class of models. Any Gaussian multivariate stationary time series can be written in a state space form, provided that the dimension of the predictor space is finite. In particular, any autoregressive moving average (ARMA) process has a state space representation and, conversely, any state space process can be expressed in an ARMA form (Akaike 1974). More details on the relation of the state space and ARMA forms are given in "Relation of ARMA and State Space Forms" later in this chapter.

Let $\mathbf{x}_t$ be the $r \times 1$ vector of observed variables, after differencing (if differencing is specified) and subtracting the sample mean. Let $\mathbf{z}_t$ be the state vector of dimension s, $s \geq r$, where the first r components of $\mathbf{z}_t$ consist of $\mathbf{x}_t$. Let the notation $\mathbf{x}_{t+k|t}$

represent the conditional expectation (or prediction) of $\mathbf{x}_{t+k}$ based on the information available at time *t*. Then the last $s - r$ elements of $\mathbf{z}_t$ consist of elements of $\mathbf{x}_{t+k|t}$, where *k*>0 is specified or determined automatically by the procedure.

There are various forms of the state space model in use. The form of the state space model used by the STATESPACE procedure is based on Akaike (1976). The model is defined by the following *state transition equation*:

$$\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$$

In the state transition equation, the $s \times s$ coefficient matrix **F** is called the *transition matrix*; it determines the dynamic properties of the model.

The $s \times r$ coefficient matrix **G** is called the *input matrix*; it determines the variance structure of the transition equation. For model identification, the first *r* rows and columns of **G** are set to an $r \times r$ identity matrix.

The input vector $\mathbf{e}_t$ is a sequence of independent normally distributed random vectors of dimension *r* with mean **0** and covariance matrix $\Sigma_{\mathbf{ee}}$. The random error $\mathbf{e}_t$ is sometimes called the innovation vector or shock vector.

In addition to the state transition equation, state space models usually include a *measurement equation* or *observation equation* that gives the observed values $\mathbf{x}_t$ as a function of the state vector $\mathbf{z}_t$. However, since PROC STATESPACE always includes the observed values $\mathbf{x}_t$ in the state vector $\mathbf{z}_t$, the measurement equation in this case merely represents the extraction of the first *r* components of the state vector.

The measurement equation used by the STATESPACE procedure is

$$\mathbf{x}_t = [\mathbf{I}_r \mathbf{0}]\mathbf{z}_t$$

where $\mathbf{I}_r$ is an $r \times r$ identity matrix. In practice, PROC STATESPACE performs the extraction of $\mathbf{x}_t$ from $\mathbf{z}_t$ without reference to an explicit measurement equation.

In summary:

$\mathbf{x}_t$	is an observation vector of dimension *r*.
$\mathbf{z}_t$	is a state vector of dimension *s*, whose first *r* elements are $\mathbf{x}_t$ and whose last $s - r$ elements are conditional prediction of future $\mathbf{x}_t$.
F	is an $s \times s$ transition matrix.
G	is an $s \times r$ input matrix, with the identity matrix $\mathbf{I}_r$ forming the first *r* rows and columns.
$\mathbf{e}_t$	is a sequence of independent normally distributed random vectors of dimension *r* with mean **0** and covariance matrix $\Sigma_{\mathbf{ee}}$.

How PROC STATESPACE Works

The design of the STATESPACE procedure closely follows the modeling strategy proposed by Akaike (1976). This strategy employs canonical correlation analysis for the automatic identification of the state space model.

Following Akaike (1976), the procedure first fits a sequence of unrestricted vector autoregressive (VAR) models and computes Akaike's information criterion (AIC) for each model. The vector autoregressive models are estimated using the sample autocovariance matrices and the Yule-Walker equations. The order of the VAR model producing the smallest Akaike information criterion is chosen as the order (number of lags into the past) to use in the canonical correlation analysis.

The elements of the state vector are then determined via a sequence of canonical correlation analyses of the sample autocovariance matrices through the selected order. This analysis computes the sample canonical correlations of the past with an increasing number of steps into the future. Variables that yield significant correlations are added to the state vector; those that yield insignificant correlations are excluded from further consideration. The importance of the correlation is judged on the basis of another information criterion proposed by Akaike. See the section "Canonical Correlation Analysis" for details. If you specify the state vector explicitly, these model identification steps are omitted.

Once the state vector is determined, the state space model is fit to the data. The free parameters in the $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{\Sigma_{ee}}$ matrices are estimated by approximate maximum likelihood. By default, the $\mathbf{F}$ and $\mathbf{G}$ matrices are unrestricted, except for identifiability requirements. Optionally, conditional least-squares estimates can be computed. You can impose restrictions on elements of the $\mathbf{F}$ and $\mathbf{G}$ matrices.

After the parameters are estimated, forecasts are produced from the fitted state space model using the Kalman filtering technique. If differencing was specified, the forecasts are integrated to produce forecasts of the original input variables.

Getting Started

The following introductory example uses simulated data for two variables X and Y. The following statements generate the X and Y series.

```
data in;
   x=10;  y=40;
   x1=0; y1=0;
   a1=0; b1=0;
   iseed=123;
   do t=-100 to 200;
      a=rannor(iseed);
      b=rannor(iseed);
      dx = 0.5*x1 + 0.3*y1 + a - 0.2*a1 - 0.1*b1;
      dy = 0.3*x1 + 0.5*y1 + b;
      x = x + dx + .25;
      y = y + dy + .25;
      if t >= 0 then output;
      x1 = dx; y1 = dy;
      a1 = a; b1 = b;
   end;
   keep t x y;
run;
```

The simulated series X and Y are shown in Figure 25.1.

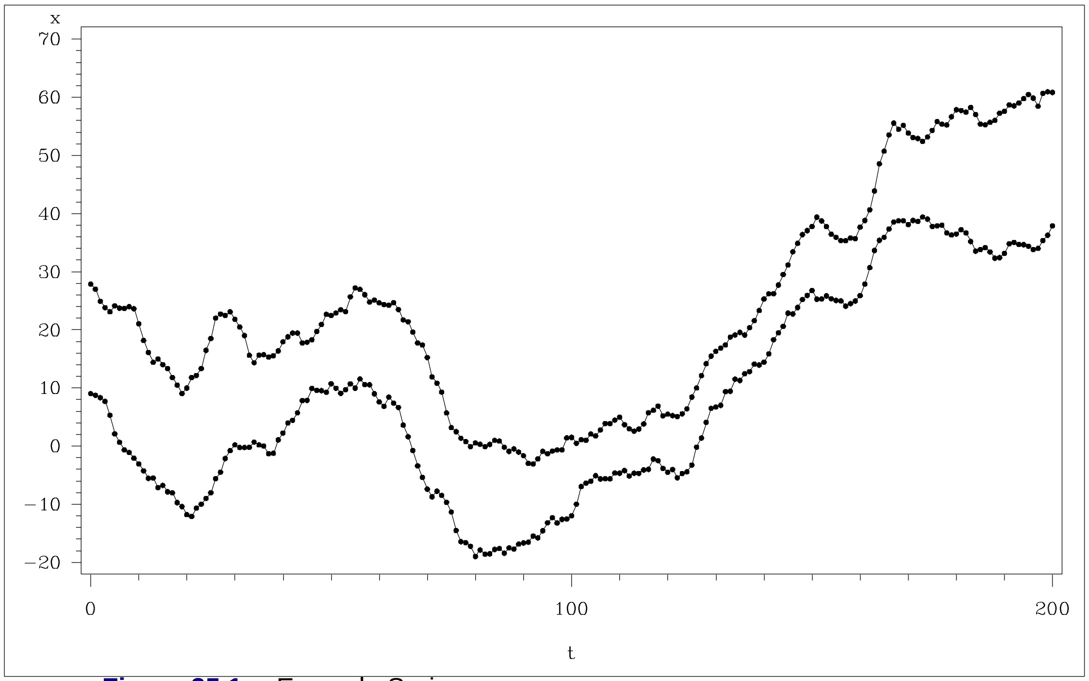


Figure 25.1. Example Series

Automatic State Space Model Selection

The STATESPACE procedure is designed to automatically select the best state space model for forecasting the series. You can specify your own model if you wish, and you can use the output from PROC STATESPACE to help you identify a state space model. However, the easiest way to use PROC STATESPACE is to let it choose the model.

Stationarity and Differencing

Although PROC STATESPACE selects the state space model automatically, it does assume that the input series are stationary. If the series are nonstationary, then the process may fail. Therefore the first step is to examine your data and test to see if differencing is required. (See the section "Stationarity and Differencing" later in this chapter for further discussion of this issue.)

The series shown in Figure 25.1 are nonstationary. In order to forecast X and Y with a state space model, you must difference them (or use some other de-trending method). If you fail to difference when needed and try to use PROC STATESPACE with nonstationary data, an inappropriate state space model may be selected, and the model estimation may fail to converge.

The following statements identify and fit a state space model for the first differences of X and Y, and forecast X and Y 10 periods ahead:

```
proc statespace data=in out=out lead=10;
   var x(1) y(1);
   id t;
run;
```

The DATA= option specifies the input data set and the OUT= option specifies the output data set for the forecasts. The LEAD= option specifies forecasting 10 observations past the end of the input data. The VAR statement specifies the variables to forecast and specifies differencing. The notation X(1) Y(1) specifies that the state space model analyzes the first differences of X and Y.

Descriptive Statistics and Preliminary Autoregressions

The first page of the printed output produced by the preceding statements is shown in Figure 25.2.

```
                            The STATESPACE Procedure

                         Number of Observations    200


                                     Standard
           Variable         Mean        Error

           x            0.144316    1.233457    Has been differenced.
                                                With period(s) = 1.
           y            0.164871    1.304358    Has been differenced.
                                                With period(s) = 1.



                            The STATESPACE Procedure

                  Information Criterion for Autoregressive Models

   Lag=0    Lag=1    Lag=2    Lag=3    Lag=4    Lag=5    Lag=6    Lag=7    Lag=8

 149.697 8.387786 5.517099 12.05986 15.36952 21.79538 24.00638 29.88874 33.55708

                                     Information
                                    Criterion for
                                   Autoregressive
                                       Models

                                   Lag=9      Lag=10

                                41.17606    47.70222


                     Schematic Representation of Correlations

   Name/Lag     0     1     2     3     4     5     6     7     8     9    10

   x           ++    ++    ++    ++    ++    ++    +.    ..    +.    +.    ..
   y           ++    ++    ++    ++    ++    +.    +.    +.    +.    ..    ..

            + is > 2*std error,  - is < -2*std error,  . is between
```

Figure 25.2. Descriptive Statistics and VAR Order Selection

Descriptive statistics are printed first, giving the number of nonmissing observations after differencing, and the sample means and standard deviations of the differenced series. The sample means are subtracted before the series are modeled (unless the NOCENTER option is specified), and the sample means are added back when the forecasts are produced.

Let X_t and Y_t be the observed values of X and Y, and let x_t and y_t be the values of X and Y after differencing and subtracting the mean difference. The series $\mathbf{x}_t$ modeled by the STATEPSPACE procedure is

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} (1-B)X_t - 0.144316 \\ (1-B)Y_t - 0.164871 \end{bmatrix}$$

where B represents the backshift operator.

After the descriptive statistics, PROC STATESPACE prints the Akaike information criterion (AIC) values for the autoregressive models fit to the series. The smallest AIC value, in this case 5.517 at lag 2, determines the number of autocovariance matrices analyzed in the canonical correlation phase.

A schematic representation of the autocorrelations is printed next. This indicates which elements of the autocorrelation matrices at different lags are significantly greater or less than 0.

The second page of the STATESPACE printed output is shown in Figure 25.3.

```
                        The STATESPACE Procedure

              Schematic Representation of Partial Autocorrelations

   Name/Lag      1     2     3     4     5     6     7     8     9    10

   x            ++    +.    ..    ..    ..    ..    ..    ..    ..    ..
   y            ++    ..    ..    ..    ..    ..    ..    ..    ..    ..

         + is > 2*std error,  - is < -2*std error,  . is between


                     Yule-Walker Estimates for Minimum AIC

                        --------Lag=1-------    --------Lag=2-------
                              x           y           x           y

         x              0.257438    0.202237    0.170812    0.133554
         y              0.292177    0.469297    -0.00537    -0.00048
```

Figure 25.3. Partial Autocorrelations and VAR Model

Figure 25.3 shows a schematic representation of the partial autocorrelations, similar to the autocorrelations shown in Figure 25.2. The selection of a second order autoregressive model by the AIC statistic looks reasonable in this case because the partial autocorrelations for lags greater than 2 are not significant.

Next, the Yule-Walker estimates for the selected autoregressive model are printed. This output shows the coefficient matrices of the vector autoregressive model at each lag.

Selected State Space Model Form and Preliminary Estimates

After the autoregressive order selection process has determined the number of lags to consider, the canonical correlation analysis phase selects the state vector. By default, output for this process is not printed. You can use the CANCORR option to print details of the canonical correlation analysis. See the section "Canonical Correlation Analysis" later in this chapter for an explanation of this process.

Once the state vector is selected the state space model is estimated by approximate maximum likelihood. Information from the canonical correlation analysis and from the preliminary autoregression is used to form preliminary estimates of the state space model parameters. These preliminary estimates are used as starting values for the iterative estimation process.

The form of the state vector and the preliminary estimates are printed next, as shown in Figure 25.4.

```
                          The STATESPACE Procedure
               Selected Statespace Form and Preliminary Estimates

                                State Vector

                x(T;T)         y(T;T)         x(T+1;T)


                        Estimate of Transition Matrix

                        0              0              1
                 0.291536       0.468762       -0.00411
                  0.24869        0.24484       0.204257


                         Input Matrix for Innovation

                               1              0
                               0              1
                        0.257438       0.202237


                       Variance Matrix for Innovation

                        0.945196       0.100786
                        0.100786       1.014703
```

Figure 25.4. Preliminary Estimates of State Space Model

Figure 25.4 first prints the state vector as X[T;T] Y[T;T] X[T+1;T]. This notation indicates that the state vector is

$$\mathbf{z}_t = \begin{bmatrix} x_{t|t} \\ y_{t|t} \\ x_{t+1|t} \end{bmatrix}$$

The notation $x_{t+1|t}$ indicates the conditional expectation or prediction of x_{t+1} based on the information available at time *t*, and $x_{t|t}$ and $y_{t|t}$ are x_t and y_t respectively.

The remainder of Figure 25.4 shows the preliminary estimates of the transition matrix $\mathbf{F}$, the input matrix $\mathbf{G}$, and the covariance matrix $\mathbf{\Sigma_{ee}}$.

Estimated State Space Model

The next page of the STATESPACE output prints the final estimates of the fitted model, as shown in Figure 25.5. This output has the same form as in Figure 25.4, but shows the maximum likelihood estimates instead of the preliminary estimates.

```
                        The STATESPACE Procedure
                  Selected Statespace Form and Fitted Model

                              State Vector

                  x(T;T)         y(T;T)         x(T+1;T)


                      Estimate of Transition Matrix

                          0              0              1
                   0.297273        0.47376       -0.01998
                     0.2301       0.228425       0.256031


                        Input Matrix for Innovation

                                 1              0
                                 0              1
                          0.257284       0.202273


                      Variance Matrix for Innovation

                          0.945188       0.100752
                          0.100752       1.014712
```

Figure 25.5. Fitted State Space Model

The estimated state space model shown in Figure 25.5 is

$$\begin{bmatrix} x_{t+1|t+1} \\ y_{t+1|t+1} \\ x_{t+2|t+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.297 & 0.474 & -0.020 \\ 0.230 & 0.228 & 0.256 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ x_{t+1|t} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.257 & 0.202 \end{bmatrix} \begin{bmatrix} e_{t+1} \\ n_{t+1} \end{bmatrix}$$

$$var \begin{bmatrix} e_{t+1} \\ n_{t+1} \end{bmatrix} = \begin{bmatrix} 0.945 & 0.101 \\ 0.101 & 1.015 \end{bmatrix}$$

The next page of the STATESPACE output lists the estimates of the free parameters in the **F** and **G** matrices with standard errors and *t* statistics, as shown in Figure 25.6.

```
                        The STATESPACE Procedure

                          Parameter Estimates

                                          Standard
                Parameter    Estimate        Error    t Value

                F(2,1)       0.297273     0.129995       2.29
                F(2,2)       0.473760     0.115688       4.10
                F(2,3)       -0.01998     0.313025      -0.06
                F(3,1)       0.230100     0.126226       1.82
                F(3,2)       0.228425     0.112978       2.02
                F(3,3)       0.256031     0.305256       0.84
                G(3,1)       0.257284     0.071060       3.62
                G(3,2)       0.202273     0.068593       2.95
```

Figure 25.6. Final Parameter Estimates

Convergence Failures

The maximum likelihood estimates are computed by an iterative nonlinear maximization algorithm, which may not converge. If the estimates fail to converge, warning messages are printed in the output.

If you encounter convergence problems, you should recheck the stationarity of the data and ensure that the specified differencing orders are correct. Attempting to fit state space models to nonstationary data is a common cause of convergence failure. You can also use the MAXIT= option to increase the number of iterations allowed, or experiment with the convergence tolerance options DETTOL= and PARMTOL=.

Forecast Data Set

The following statements print the output data set. The WHERE statement excludes the first 190 observations from the output, so that only the forecasts and the last 10 actual observations are printed.

```
proc print data=out;
   id t;
   where t > 190;
run;
```

The PROC PRINT output is shown in Figure 25.7.

```
  t       x       FOR1      RES1       STD1       y       FOR2      RES2       STD2

191  34.8159  33.6299   1.18600  0.97221  58.7189  57.9916   0.72728  1.00733
192  35.0656  35.6598  -0.59419  0.97221  58.5440  59.7718  -1.22780  1.00733
193  34.7034  35.5530  -0.84962  0.97221  59.0476  58.5723   0.47522  1.00733
194  34.6626  34.7597  -0.09707  0.97221  59.7774  59.2241   0.55330  1.00733
195  34.4055  34.8322  -0.42664  0.97221  60.5118  60.1544   0.35738  1.00733
196  33.8210  34.6053  -0.78434  0.97221  59.8750  60.8260  -0.95102  1.00733
197  34.0164  33.6230   0.39333  0.97221  58.4698  59.4502  -0.98046  1.00733
198  35.3819  33.6251   1.75684  0.97221  60.6782  57.9167   2.76150  1.00733
199  36.2954  36.0528   0.24256  0.97221  60.9692  62.1637  -1.19450  1.00733
200  37.8945  37.1431   0.75142  0.97221  60.8586  61.4085  -0.54984  1.00733
201        .  38.5068         .  0.97221        .  61.3161         .  1.00733
202        .  39.0428         .  1.59125        .  61.7509         .  1.83678
203        .  39.4619         .  2.28028        .  62.1546         .  2.62366
204        .  39.8284         .  2.97824        .  62.5099         .  3.38839
205        .  40.1474         .  3.67689        .  62.8275         .  4.12805
206        .  40.4310         .  4.36299        .  63.1139         .  4.84149
207        .  40.6861         .  5.03040        .  63.3755         .  5.52744
208        .  40.9185         .  5.67548        .  63.6174         .  6.18564
209        .  41.1330         .  6.29673        .  63.8435         .  6.81655
210        .  41.3332         .  6.89383        .  64.0572         .  7.42114
```

Figure 25.7. OUT= Data Set Produced by PROC STATESPACE

The OUT= data set produced by PROC STATESPACE contains the VAR and ID statement variables. In addition, for each VAR statement variable, the OUT= data set contains the variables FOR*i*, RES*i*, and STD*i*. These variables contain the predicted values, residuals, and forecast standard errors for the *i*th variable in the VAR statement list. In this case, X is listed first in the VAR statement, so FOR1 contains the forecasts of X, while FOR2 contains the forecasts of Y.

The following statements plot the forecasts and actuals for the series.

```
proc gplot data=out;
   plot for1*t=1 for2*t=1 x*t=2 y*t=2 /
        overlay href=200.5;
   symbol1 v=circle i=join;
   symbol2 v=star i=none;
   where t > 150;
run;
```

The forecast plot is shown in Figure 25.8. The last 50 observations are also plotted to provide context, and a reference line is drawn between the historical and forecast periods. The actual values are plotted with asterisks.

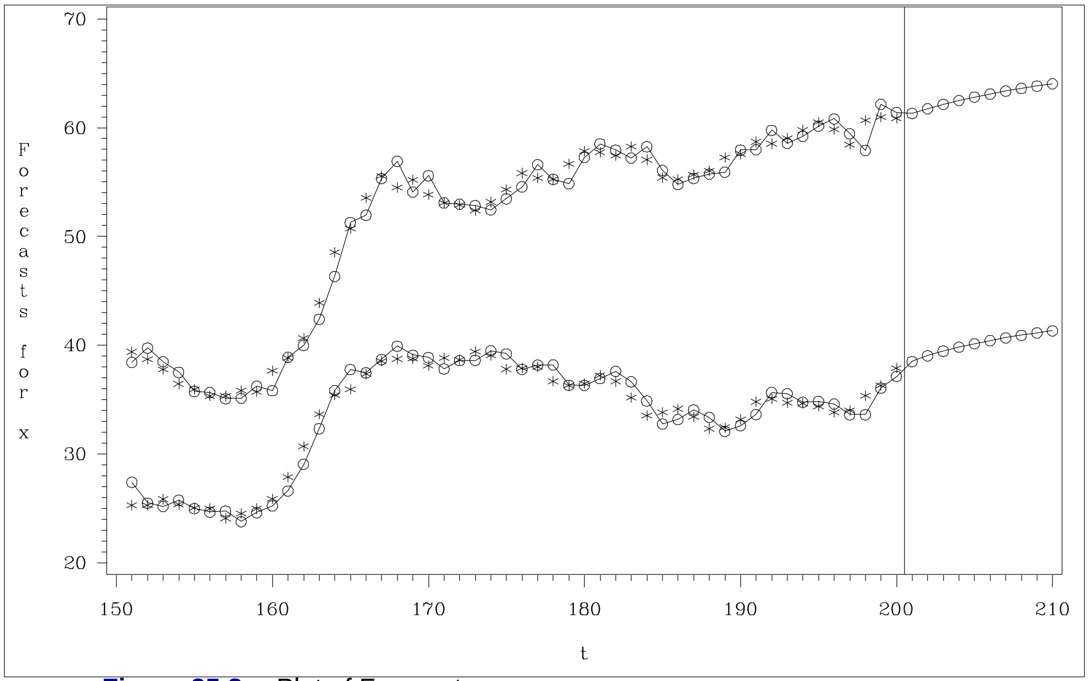


Figure 25.8. Plot of Forecasts

Controlling Printed Output

By default, the STATESPACE procedure produces a large amount of printed output. The NOPRINT option suppresses all printed output. You can suppress the printed output for the autoregressive model selection process with the PRINTOUT=NONE option. The descriptive statistics and state space model estimation output are still printed when PRINTOUT=NONE is specified. You can produce more detailed output with the PRINTOUT=LONG option and by specifying the printing control options CANCORR, COVB, and PRINT.

Specifying the State Space Model

Instead of allowing the STATESPACE procedure to select the model automatically, you can use FORM and RESTRICT statements to specify a state space model.

Specifying the State Vector

Use the FORM statement to control the form of the state vector. You can use this feature to force PROC STATESPACE to estimate and forecast a model different from the model it would select automatically. You can also use this feature to reestimate the automatically selected model (possibly with restrictions) without repeating the canonical correlation analysis.

The FORM statement specifies the number of lags of each variable to include in the state vector. For example, the statement FORM X 3; forces the state vector to include $x_{t|t}$, $x_{t+1|t}$, and $x_{t+2|t}$. The following statement specifies the state vector $(x_{t|t}, y_{t|t}, x_{t+1|t})$, which is the same state vector selected in the preceding example:

```
   form x 2 y 1;
```

You can specify the form for only some of the variables and allow PROC STATESPACE to select the form for the other variables. If only some of the variables are specified in the FORM statement, canonical correlation analysis is used to determine the number of lags included in the state vector for the remaining variables not specified by the FORM statement. If the FORM statement includes specifications for all the variables listed in the VAR statement, the state vector is completely defined and the canonical correlation analysis is not performed.

Restricting the F and G matrices

After you know the form of the state vector, you can use the RESTRICT statement to fix some parameters in the **F** and **G** matrices to specified values. One use of this feature is to remove insignificant parameters by restricting them to 0.

In the introductory example shown in the preceding section, the F[2,3] parameter is not significant. (The parameters estimation output shown in Figure 25.6 gives the *t* statistic for F[2,3] as -0.06. F[3,3] and F[3,1] also have low significance with $t < 2$.)

The following statements reestimate this model with F[2,3] restricted to 0. The FORM statement is used to specify the state vector and thus bypass the canonical correlation analysis.

```
   proc statespace data=in out=out lead=10;
      var x(1) y(1);
      id t;
      form x 2 y 1;
      restrict f(2,3)=0;
   run;
```

The final estimates produced by these statements are shown in Figure 25.9.

```
                    The STATESPACE Procedure
          Selected Statespace Form and Fitted Model

                         State Vector

          x(T;T)         y(T;T)         x(T+1;T)


                 Estimate of Transition Matrix

                   0             0             1
            0.290051      0.467468             0
            0.227051      0.226139       0.26436


                  Input Matrix for Innovation

                          1             0
                          0             1
                   0.256826      0.202022


                 Variance Matrix for Innovation

                   0.945175      0.100696
                   0.100696      1.014733
```

```
                    The STATESPACE Procedure

                      Parameter Estimates

                                    Standard
         Parameter     Estimate        Error    t Value

         F(2,1)        0.290051     0.063904       4.54
         F(2,2)        0.467468     0.060430       7.74
         F(3,1)        0.227051     0.125221       1.81
         F(3,2)        0.226139     0.111711       2.02
         F(3,3)        0.264360     0.299537       0.88
         G(3,1)        0.256826     0.070994       3.62
         G(3,2)        0.202022     0.068507       2.95
```

Figure 25.9. Results using RESTRICT Statement

Syntax

The STATESPACE procedure uses the following statements:

PROC STATESPACE *options*;
  BY *variable . . .* ;
  FORM *variable value . . .* ;
  ID *variable*;
  INITIAL **F**(*row,column*)=*value . . .* **G**(*row,column*)=*value ...* ;
  RESTRICT **F**(*row,column*)=*value . . .* **G**(*row,column*)=*value ...* ;
  VAR *variable (difference, difference, . . .) ...* ;

Functional Summary

The statements and options used by PROC STATESPACE are summarized in the following table.

Description	Statement	Option
Input Data Set Options		
specify the input data set	PROC STATESPACE	DATA=
prevent subtraction of sample mean	PROC STATESPACE	NOCENTER
specify the ID variable	ID	
specify the observed series and differencing	VAR	
Options for Autoregressive Estimates		
specify the maximum order	PROC STATESPACE	ARMAX=
specify maximum lag for autocovariances	PROC STATESPACE	LAGMAX=
output only minimum AIC model	PROC STATESPACE	MINIC
specify the amount of detail printed	PROC STATESPACE	PRINTOUT=
write preliminary AR models to a data set	PROC STATESPACE	OUTAR=
Options for Canonical Correlation Analysis		
print the sequence of canonical correlations	PROC STATESPACE	CANCORR
specify upper limit of dimension of state vector	PROC STATESPACE	DIMMAX=
specify the minimum number of lags	PROC STATESPACE	PASTMIN=
specify the multiplier of the degrees of freedom	PROC STATESPACE	SIGCORR=
Options for State Space Model Estimation		
specify starting values	INITIAL	
print covariance matrix of parameter estimates	PROC STATESPACE	COVB
specify the convergence criterion	PROC STATESPACE	DETTOL=
specify the convergence criterion	PROC STATESPACE	PARMTOL=

Description	Statement	Option
print the details of the iterations	PROC STATESPACE	ITPRINT
specify an upper limit of the number of lags	PROC STATESPACE	KLAG=
specify maximum number of iterations allowed	PROC STATESPACE	MAXIT=
suppress the final estimation	PROC STATESPACE	NOEST
write the state space model parameter estimates to an output data set	PROC STATESPACE	OUTMODEL=
use conditional least squares for final estimates	PROC STATESPACE	RESIDEST
specify criterion for testing for singularity	PROC STATESPACE	SINGULAR=
Options for Forecasting		
start forecasting before end of the input data	PROC STATESPACE	BACK=
specify the time interval between observations	PROC STATESPACE	INTERVAL=
specify multiple periods in the time series	PROC STATESPACE	INTPER=
specify how many periods to forecast	PROC STATESPACE	LEAD=
specify the output data set for forecasts	PROC STATESPACE	OUT=
print forecasts	PROC STATESPACE	PRINT
Options to Specify the State Space Model		
specify the state vector	FORM	
specify the parameter values	RESTRICT	
BY Groups		
specify BY-group processing	BY	
Printing		
suppresses all printed output	NOPRINT	

PROC STATESPACE Statement

PROC STATESPACE *options***;**

The following options can be specified in the PROC STATESPACE statement.

Printing Options

NOPRINT

suppresses all printed output.

Input Data Options

DATA= *SAS-data-set*

specifies the name of the SAS data set to be used by the procedure. If the DATA= option is omitted, the most recently created SAS data set is used.

LAGMAX= *k*

specifies the number of lags for which the sample autocovariance matrix is computed. The LAGMAX= option controls the number of lags printed in the schematic representation of the autocorrelations.

The sample autocovariance matrix of lag *i*, denoted as $\mathbf{C}_i$, is computed as

$$\mathbf{C}_i = \frac{1}{N-1}\sum_{t=1+i}^{N} \mathbf{x}_t\mathbf{x}'_{t-i}$$

where $\mathbf{x}_t$ is the differenced and centered data and N is the number of observations. (If the NOCENTER option is specified, 1 is not subtracted from N.) LAGMAX= *k* specifies that $\mathbf{C}_0$ through $\mathbf{C}_k$ are computed. The default is LAGMAX=10.

NOCENTER

prevents subtraction of the sample mean from the input series (after any specified differencing) before the analysis.

Options for Preliminary Autoregressive Models

ARMAX= *n*

specifies the maximum order of the preliminary autoregressive models. The ARMAX= option controls the autoregressive orders for which information criteria are printed, and controls the number of lags printed in the schematic representation of partial autocorrelations. The default is ARMAX=10. See "Preliminary Autoregressive Models" later in this chapter for details.

MINIC

writes to the OUTAR= data set only the preliminary Yule-Walker estimates for the VAR model producing the minimum AIC. See "OUTAR= Data Set" later in this chapter for details.

OUTAR= *SAS-data-set*

writes the Yule-Walker estimates of the preliminary autoregressive models to a SAS data set. See "OUTAR= Data Set" later in this chapter for details.

PRINTOUT= SHORT | LONG | NONE

determines the amount of detail printed. PRINTOUT=LONG prints the lagged covariance matrices, the partial autoregressive matrices, and estimates of the residual covariance matrices from the sequence of autoregressive models. PRINTOUT=NONE suppresses the output for the preliminary autoregressive models. The descriptive statistics and state space model estimation output are still printed when PRINTOUT=NONE is specified. PRINTOUT=SHORT is the default.

Canonical Correlation Analysis Options

CANCORR
prints the canonical correlations and information criterion for each candidate state vector considered. See "Canonical Correlation Analysis" later in this chapter for details.

DIMMAX= *n*
specifies the upper limit to the dimension of the state vector. The DIMMAX= option can be used to limit the size of the model selected. The default is DIMMAX=10.

PASTMIN= *n*
specifies the minimum number of lags to include in the canonical correlation analysis. The default is PASTMIN=0. See "Canonical Correlation Analysis" later in this chapter for details.

SIGCORR= *value*
specifies the multiplier of the degrees of freedom for the penalty term in the information criterion used to select the state space form. The default is SIGCORR=2. The larger the value of the SIGCORR= option, the smaller the state vector tends to be. Hence, a large value causes a simpler model to be fit. See "Canonical Correlations Analysis" later in this chapter for details.

State Space Model Estimation Options

COVB
prints the inverse of the observed information matrix for the parameter estimates. This matrix is an estimate of the covariance matrix for the parameter estimates.

DETTOL= *value*
specifies the convergence criterion. The DETTOL= and PARMTOL= option values are used together to test for convergence of the estimation process. If, during an iteration, the relative change of the parameter estimates is less than the PARMTOL= value and the relative change of the determinant of the innovation variance matrix is less than the DETTOL= value, then iteration ceases and the current estimates are accepted. The default is DETTOL=1E-5.

ITPRINT
prints the iterations during the estimation process.

KLAG= *n*
sets an upper limit for the number of lags of the sample autocovariance matrix used in computing the approximate likelihood function. If the data have a strong moving average character, a larger KLAG= value may be necessary to obtain good estimates. The default is KLAG=15. See "Parameter Estimation" later in this chapter for details.

MAXIT= *n*
sets an upper limit to the number of iterations in the maximum likelihood or conditional least-squares estimation. The default is MAXIT=50.

NOEST

suppresses the final maximum likelihood estimation of the selected model.

OUTMODEL= *SAS-data-set*

writes the parameter estimates and their standard errors to a SAS data set. See "OUTMODEL= Data Set" later in this chapter for details.

PARMTOL= *value*

specifies the convergence criterion. The DETTOL= and PARMTOL= option values are used together to test for convergence of the estimation process. If, during an iteration, the relative change of the parameter estimates is less than the PARMTOL= value and the relative change of the determinant of the innovation variance matrix is less than the DETTOL= value, then iteration ceases and the current estimates are accepted. The default is PARMTOL=.001.

RESIDEST

computes the final estimates using conditional least squares on the raw data. This type of estimation may be more stable than the default maximum likelihood method but is usually more computationally expensive. See "Parameter Estimation" later in this chapter for details of the conditional least squares method.

SINGULAR= *value*

specifies the criterion for testing for singularity of a matrix. A matrix is declared singular if a scaled pivot is less than the SINGULAR= value when sweeping the matrix. The default is SINGULAR=1E-7.

Forecasting Options

BACK= *n*

starts forecasting *n* periods before the end of the input data. The BACK= option value must not be greater than the number of observations. The default is BACK=0.

INTERVAL= *interval*

specifies the time interval between observations. The INTERVAL= value is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data. See Chapter 3, "Date Intervals, Formats, and Functions," for details on the INTERVAL= values allowed.

INTPER= *n*

specifies that each input observation corresponds to *n* time periods. For example, the options INTERVAL=MONTH and INTPER=2 specify bimonthly data and are equivalent to specifying INTERVAL=MONTH2. If the INTERVAL= option is not specified, the INTPER= option controls the increment used to generate ID values for the forecast observations. The default is INTPER=1.

LEAD= *n*

specifies how many forecast observations are produced. The forecasts start at the point set by the BACK= option. The default is LEAD=0, which produces no forecasts.

OUT= *SAS-data-set*
writes the residuals, actual values, forecasts, and forecast standard errors to a SAS data set. See "OUT= Data Set" later in this chapter for details.

PRINT
prints the forecasts.

BY Statement

BY *variable ...* **;**

A BY statement can be used with the STATESPACE procedure to obtain separate analyses on observations in groups defined by the BY variables.

FORM Statement

FORM *variable value ...* **;**

The FORM statement specifies the number of times a variable is included in the state vector. Values can be specified for any variable listed in the VAR statement. If a value is specified for each variable in the VAR statement, the state vector for the state space model is entirely specified, and automatic selection of the state space model is not performed.

The FORM statement forces the state vector, $\mathbf{z}_t$, to contain a specific variable a given number of times. For example, if Y is one of the variables in $\mathbf{x}_t$, then the statement

```
form y 3;
```

forces the state vector to contain $Y_t, Y_{t+1|t}$, and $Y_{t+2|t}$, possibly along with other variables.

The following statements illustrate the use of the FORM statement:

```
proc statespace data=in;
   var x y;
   form x 3 y 2;
run;
```

These statements fit a state space model with the following state vector:

$$\mathbf{z_t} = \begin{bmatrix} x_{t|t} \\ y_{t|t} \\ x_{t+1|t} \\ y_{t+1|t} \\ x_{t+2|t} \end{bmatrix}$$

ID Statement

ID *variable***;**

The ID statement specifies a variable that identifies observations in the input data set. The variable specified in the ID statement is included in the OUT= data set. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= and INTPER= options.

INITIAL Statement

INITIAL F *(row,column)***=** *value* **...** **G***(row, column)***=** *value* **... ;**

The INITIAL statement gives initial values to the specified elements of the **F** and **G** matrices. These initial values are used as starting values for the iterative estimation.

Parts of the **F** and **G** matrices represent fixed structural identities. If an element specified is a fixed structural element instead of a free parameter, the corresponding initialization is ignored.

The following is an example of an INITIAL statement:

```
initial f(3,2)=0 g(4,1)=0 g(5,1)=0;
```

RESTRICT Statement

RESTRICT F*(row,column)***=** *value* **...** **G***(row,column)***=** *value* **... ;**

The RESTRICT statement restricts the specified elements of the **F** and **G** matrices to the specified values.

To use the restrict statement you need to know the form of the model. Either specify the form of the model with the FORM statement, or do a preliminary run, perhaps with the NOEST option, to find the form of the model that PROC STATESPACE selects for the data.

The following is an example of a RESTRICT statement:

```
restrict f(3,2)=0 g(4,1)=0 g(5,1)=0 ;
```

Parts of the **F** and **G** matrices represent fixed structural identities. If a restriction is specified for an element that is a fixed structural element instead of a free parameter, the restriction is ignored.

VAR Statement

VAR *variable (difference, difference, ...) ...* **;**

The VAR statement specifies the variables in the input data set to model and forecast. The VAR statement also specifies differencing of the input variables. The VAR statement is required.

Differencing is specified by following the variable name with a list of difference periods separated by commas. See the section "Stationarity and Differencing" for more information on differencing of input variables.

The order in which variables are listed in the VAR statement controls the order in which variables are included in the state vector. Usually, potential inputs should be listed before potential outputs.

For example, assuming the input data are monthly, the following VAR statement specifies modeling and forecasting of the one period and seasonal second difference of X and Y:

```
var x(1,12) y(1,12);
```

In this example, the vector time series analyzed is

$$\mathbf{x}_t = \begin{bmatrix} (1-B)(1-B^{12})X_t - \overline{x} \\ (1-B)(1-B^{12})Y_t - \overline{y} \end{bmatrix}$$

where B represents the back shift operator, and $\overline{x}$ and $\overline{y}$ represent the means of the differenced series. If the NOCENTER option is specified the mean differences are not subtracted.

Details

Missing Values

The STATESPACE procedure does not support missing values. The procedure uses the first contiguous group of observations with no missing values for any of the VAR statement variables. Observations at the beginning of the data set with missing values for any VAR statement variable are not used or included in the output data set.

Stationarity and Differencing

The state space model used by the STATESPACE procedure assumes that the time series are stationary. Hence, the data should be checked for stationarity. One way to check for stationarity is to plot the series. A graph of series over time can show a time trend or variability changes.

You can also check stationarity by using the sample autocorrelation functions displayed by the ARIMA procedure. The autocorrelation functions of nonstationary series tend to decay slowly. See Chapter 11, "The ARIMA Procedure," for more information.

Another alternative is to use the STATIONARITY= option on the IDENTIFY statement in PROC ARIMA to apply Dickey-Fuller tests for unit roots in the time series. See Chapter 11, "The ARIMA Procedure," for more information on Dickey-Fuller unit root tests.

The most popular way to transform a nonstationary series to stationarity is by differencing. Differencing of the time series is specified in the VAR statement. For example, to take a simple first difference of the series X, use this statement:

```
   var x(1);
```

In this example, the change in X from one period to the next is analyzed. When the series has a seasonal pattern, differencing at a period equal to the length of the seasonal cycle may be desirable. For example, suppose the variable X is measured quarterly and shows a seasonal cycle over the year. You can use the following statement to analyze the series of changes from the same quarter in the previous year:

```
   var x(4);
```

To difference twice, add another differencing period to the list. For example, the following statement analyzes the series of second differences $(X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}$:

```
   var x(1,1);
```

The following statement analyzes the seasonal second difference series.

```
var x(1,4);
```

The series modeled is the 1-period difference of the 4-period difference: $(X_t - X_{t-4}) - (X_{t-1} - X_{t-5}) = X_t - X_{t-1} - X_{t-4} + X_{t-5}$.

Another way to obtain stationary series is to use a regression on time to de-trend the data. If the time series has a deterministic linear trend, regressing the series on time produces residuals that should be stationary. The following statements write residuals of X and Y to the variable RX and RY in the output data set DETREND.

```
data a;
   set a;
   t=_n_;
run;

proc reg data=a;
   model x y = t;
   output out=detrend r=rx ry;
run;
```

You then use PROC STATESPACE to forecast the de-trended series RX and RY. A disadvantage of this method is that you need to add the trend back to the forecast series in an additional step. A more serious disadvantage of the de-trending method is that it assumes a deterministic trend. In practice, most time series appear to have a stochastic rather than a deterministic trend. Differencing is a more flexible and often more appropriate method.

There are several other methods to handle nonstationary time series. For more information and examples, refer to Brockwell and Davis (1991).

Preliminary Autoregressive Models

After computing the sample autocovariance matrices, PROC STATESPACE fits a sequence of vector autoregressive models. These preliminary autoregressive models are used to estimate the autoregressive order of the process and limit the order of the autocovariances considered in the state vector selection process.

Yule-Walker Equations for Forward and Backward Models

Unlike a univariate autoregressive model, a multivariate autoregressive model has different forms, depending on whether the present observation is being predicted from the past observations or from the future observations.

Let $\mathbf{x}_t$ be the *r*-component stationary time series given by the VAR statement after differencing and subtracting the vector of sample means. (If the NOCENTER option is specified, the mean is not subtracted.) Let *n* be the number of observations of $\mathbf{x}_t$ from the input data set.

Let $\mathbf{e}_t$ be a vector white noise sequence with mean vector **0** and variance matrix $\mathbf{\Sigma}_p$, and let $\mathbf{n}_t$ be a vector white noise sequence with mean vector **0** and variance matrix $\mathbf{\Omega}_p$. Let *p* be the order of the vector autoregressive model for $\mathbf{x}_t$.

The forward autoregressive form based on the past observations is written as follows:

$$\mathbf{x}_t = \sum_{i=1}^{p} \mathbf{\Phi}_i^p \mathbf{x}_{t-i} + \mathbf{e}_t$$

The backward autoregressive form based on the future observations is written as follows:

$$\mathbf{x}_t = \sum_{i=1}^{p} \mathbf{\Psi}_i^p \mathbf{x}_{t+i} + \mathbf{n}_t$$

Letting E denote the expected value operator, the autocovariance sequence for the $\mathbf{x}_t$ series, $\mathbf{\Gamma}_i$, is

$$\mathbf{\Gamma}_i = E\mathbf{x}_t\mathbf{x}'_{t-i}$$

The Yule-Walker equations for the autoregressive model that matches the first *p* elements of the autocovariance sequence are

$$\begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}_1 & \cdots & \mathbf{\Gamma}_{p-1} \\ \mathbf{\Gamma}'_1 & \mathbf{\Gamma}_0 & \cdots & \mathbf{\Gamma}_{p-2} \\ \vdots & \vdots & & \vdots \\ \mathbf{\Gamma}'_{p-1} & \mathbf{\Gamma}'_{p-2} & \cdots & \mathbf{\Gamma}_0 \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_1^p \\ \mathbf{\Phi}_2^p \\ \vdots \\ \mathbf{\Phi}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma}_1 \\ \mathbf{\Gamma}_2 \\ \vdots \\ \mathbf{\Gamma}_p \end{bmatrix}$$

and

$$\begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}'_1 & \cdots & \mathbf{\Gamma}'_{p-1} \\ \mathbf{\Gamma}_1 & \mathbf{\Gamma}_0 & \cdots & \mathbf{\Gamma}'_{p-2} \\ \vdots & \vdots & & \vdots \\ \mathbf{\Gamma}_{p-1} & \mathbf{\Gamma}_{p-2} & \cdots & \mathbf{\Gamma}_0 \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_1^p \\ \mathbf{\Psi}_2^p \\ \vdots \\ \mathbf{\Psi}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma}'_1 \\ \mathbf{\Gamma}'_2 \\ \vdots \\ \mathbf{\Gamma}'_p \end{bmatrix}$$

Here $\mathbf{\Phi}_i^p$ are the coefficient matrices for the past observation form of the vector autoregressive model, and $\mathbf{\Psi}_i^p$ are the coefficient matrices for the future observation form. More information on the Yule-Walker equations in the multivariate setting can be found in Whittle (1963) and Ansley and Newbold (1979).

The innovation variance matrices for the two forms can be written as follows:

$$\mathbf{\Sigma}_p = \mathbf{\Gamma}_0 - \sum_{i=1}^{p} \mathbf{\Phi}_i^p \mathbf{\Gamma}'_i$$

$$\mathbf{\Omega}_p = \mathbf{\Gamma}_0 - \sum_{i=1}^{p} \mathbf{\Psi}_i^p \mathbf{\Gamma}_i$$

The autoregressive models are fit to the data using the preceding Yule-Walker equations with $\mathbf{\Gamma}_i$ replaced by the sample covariance sequence $\mathbf{C_i}$. The covariance matrices are calculated as

$$\mathbf{C}_i = \frac{1}{N-1} \sum_{t=i+1}^{N} \mathbf{x}_t \mathbf{x}'_{t-i}$$

Let $\widehat{\mathbf{\Phi}}_p$, $\widehat{\mathbf{\Psi}}_p$, $\widehat{\mathbf{\Sigma}}_p$, and $\widehat{\mathbf{\Omega}}_p$ represent the Yule-Walker estimates of $\mathbf{\Phi}_p$, $\mathbf{\Psi}_p$, $\mathbf{\Sigma}_p$, and $\mathbf{\Omega}_p$ respectively. These matrices are written to an output data set when the OUTAR= option is specified.

When the PRINTOUT=LONG option is specified, the sequence of matrices $\widehat{\mathbf{\Sigma}}_p$ and the corresponding correlation matrices are printed. The sequence of matrices $\widehat{\mathbf{\Sigma}}_p$ is used to compute Akaike information criteria for selection of the autoregressive order of the process.

Akaike Information Criterion

The Akaike information criterion, or AIC, is defined as -2(*maximum of log likelihood*)+2(*number of parameters*). Since the vector autoregressive models are estimates from the Yule-Walker equations, not by maximum likelihood, the exact likelihood values are not available for computing the AIC. However, for the vector autoregressive model the maximum of the log likelihood can be approximated as

$$\ln(L) \approx -\frac{n}{2}\ln(|\widehat{\mathbf{\Sigma}}_p|)$$

Thus, the AIC for the order *p* model is computed as

$$AIC_p = n\ln(|\widehat{\mathbf{\Sigma}}_p|) + 2pr^2$$

You can use the printed AIC array to compute a likelihood ratio test of the autoregressive order. The log-likelihood ratio test statistic for testing the order *p* model against the order $p-1$ model is

$$-n\ln(|\widehat{\mathbf{\Sigma}}_p|) + n\ln(|\widehat{\mathbf{\Sigma}}_{p-1}|)$$

This quantity is asymptotically distributed as a χ^2 with r^2 degrees of freedom if the series is autoregressive of order $p-1$. It can be computed from the AIC array as

$$AIC_{p-1} - AIC_p + 2r^2$$

You can evaluate the significance of these test statistics with the PROBCHI function in a SAS DATA step, or with a χ^2 table.

Determining the Autoregressive Order

Although the autoregressive models can be used for prediction, their primary value is to aid in the selection of a suitable portion of the sample covariance matrix for use in computing canonical correlations. If the multivariate time series $\mathbf{x}_t$ is of autoregressive order *p*, then the vector of past values to lag *p* is considered to contain essentially all the information relevant for prediction of future values of the time series.

By default, PROC STATESPACE selects the order, *p*, producing the autoregressive model with the smallest AIC_p. If the value *p* for the minimum AIC_p is less than the value of the PASTMIN= option, then *p* is set to the PASTMIN= value. Alternatively, you can use the ARMAX= and PASTMIN= options to force PROC STATESPACE to use an order you select.

Significance Limits for Partial Autocorrelations

The STATESPACE procedure prints a schematic representation of the partial autocorrelation matrices indicating which partial autocorrelations are significantly greater or significantly less than 0. Figure 25.10 shows an example of this table.

```
                              The STATESPACE Procedure

                 Schematic Representation of Partial Autocorrelations

      Name/Lag      1     2     3     4     5     6     7     8     9    10

      x             ++    +.    ..    ..    ..    ..    ..    ..    ..    ..
      y             ++    ..    ..    ..    ..    ..    ..    ..    ..    ..

            + is > 2*std error,  - is < -2*std error,  . is between
```

Figure 25.10. Significant Partial Autocorrelations

The partial autocorrelations are from the sample partial autoregressive matrices $\hat{\mathbf{\Phi}}^p_p$. The standard errors used for the significance limits of the partial autocorrelations are computed from the sequence of matrices $\mathbf{\Sigma}_p$ and $\mathbf{\Omega}_p$.

Under the assumption that the observed series arises from an autoregressive process of order $p - 1$, the *p*th sample partial autoregressive matrix $\hat{\mathbf{\Phi}}^p_p$ has an asymptotic variance matrix $\frac{1}{n}\mathbf{\Omega}_p^{-1}\otimes\mathbf{\Sigma}_p$.

The significance limits for $\hat{\mathbf{\Phi}}^p_p$ used in the schematic plot of the sample partial autoregressive sequence are derived by replacing $\mathbf{\Omega}_p$ and $\mathbf{\Sigma}_p$ with their sample estimators to produce the variance estimate, as follows:

$$\widehat{Var}\left(\hat{\mathbf{\Phi}}^p_p\right) = \left(\frac{1}{n - rp}\right)\hat{\mathbf{\Omega}}_p^{-1}\otimes\hat{\mathbf{\Sigma}}_p$$

Canonical Correlation Analysis

Given the order *p*, let $\mathbf{p}_t$ be the vector of current and past values relevant to prediction of $\mathbf{x}_{t+1}$:

$$\mathbf{p}_t = (\mathbf{x}'_t, \mathbf{x}'_{t-1}, \cdots, \mathbf{x}'_{t-p})'$$

Let $\mathbf{f}_t$ be the vector of current and future values:

$$\mathbf{f}_t = (\mathbf{x}'_t, \mathbf{x}'_{t+1}, \cdots, \mathbf{x}'_{t+p})'$$

In the canonical correlation analysis, consider submatrices of the sample covariance matrix of $\mathbf{p}_t$ and $\mathbf{f}_t$. This covariance matrix, $\mathbf{V}$, has a block Hankel form:

$$\mathbf{V} = \begin{bmatrix} \mathbf{C}_0 & \mathbf{C}'_1 & \mathbf{C}'_2 & \cdots & \mathbf{C}'_p \\ \mathbf{C}'_1 & \mathbf{C}'_2 & \mathbf{C}'_3 & \cdots & \mathbf{C}'_{p+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{C}'_p & \mathbf{C}'_{p+1} & \mathbf{C}'_{p+2} & \cdots & \mathbf{C}'_{2p} \end{bmatrix}$$

State Vector Selection Process

The canonical correlation analysis forms a sequence of potential state vectors, $\mathbf{z}_t^j$. Examine a sequence, $\mathbf{f}_t^j$, of subvectors of $\mathbf{f}_t$, and form the submatrix, $\mathbf{V}^j$, consisting of the rows and columns of $\mathbf{V}$ corresponding to the components of $\mathbf{f}_t^j$, and compute its canonical correlations.

The smallest canonical correlation of $\mathbf{V}^j$ is then used in the selection of the components of the state vector. The selection process is described in the following. For more details about this process, refer to Akaike (1976).

In the following discussion, the notation $\mathbf{x}_{t+k|t}$ denotes the wide sense conditional expectation (best linear predictor) of $\mathbf{x}_{t+k}$, given all $\mathbf{x}_s$ with *s* less than or equal to *t*. In the notation $x_{i,t+1}$, the first subscript denotes the *i*th component of $\mathbf{x}_{t+1}$.

The initial state vector $\mathbf{z}_t^1$ is set to $\mathbf{x}_t$. The sequence $\mathbf{f}_t^j$ is initialized by setting

$$\mathbf{f}_t^1 = (\mathbf{z}_t^{1'}, x_{1,t+1|t})' = (\mathbf{x}'_t, x_{1,t+1|t})'$$

That is, start by considering whether to add $x_{1,t+1|t}$ to the initial state vector $\mathbf{z}_t^1$.

The procedure forms the submatrix $\mathbf{V}^1$ corresponding to $\mathbf{f}_t^1$ and computes its canonical correlations. Denote the smallest canonical correlation of $\mathbf{V}^1$ as ρ_{min}. If ρ_{min} is significantly greater than 0, $x_{1,t+1|t}$ is added to the state vector.

If the smallest canonical correlation of $\mathbf{V}^1$ is not significantly greater than 0, then a linear combination of $\mathbf{f}_t^1$ is uncorrelated with the past, $\mathbf{p}_t$. Assuming that the determinant of $\mathbf{C}_0$ is not 0, (that is, no input series is a constant), you can take the coefficient of $x_{1,t+1|t}$ in this linear combination to be 1. Denote the coefficients of $\mathbf{z}_t^1$ in this linear combination as ℓ. This gives the relationship:

$$x_{1,t+1|t} = \ell' \mathbf{x}_t$$

Therefore, the current state vector already contains all the past information useful for predicting $x_{1,t+1}$ and any greater leads of $x_{1,t}$. The variable $x_{1,t+1|t}$ is not added to the state vector, nor are any terms $x_{1,t+k|t}$ considered as possible components of the state vector. The variable x_1 is no longer active for state vector selection.

The process described for $x_{1,t+1|t}$ is repeated for the remaining elements of $\mathbf{f}_t$. The next candidate for inclusion in the state vector is the next component of $\mathbf{f}_t$ corresponding to an active variable. Components of $\mathbf{f}_t$ corresponding to inactive variables that produced a zero ρ_{min} in a previous step are skipped.

Denote the next candidate as $x_{l,t+k|t}$. The vector $\mathbf{f}_t^j$ is formed from the current state vector and $x_{l,t+k|t}$ as follows:

$$\mathbf{f}_t^j = (\mathbf{z}_t^{j'}, x_{l,t+k|t})'$$

The matrix $\mathbf{V}^j$ is formed from $\mathbf{f}_t^j$ and its canonical correlations are computed. The smallest canonical correlation of $\mathbf{V}^j$ is judged to be either greater than or equal to 0. If it is judged to be greater than 0, $x_{l,t+k|t}$ is added to the state vector. If it is judged to be 0, then a linear combination of $\mathbf{f}_t^j$ is uncorrelated with the $\mathbf{p}_t$, and the variable x_l is now inactive.

The state vector selection process continues until no active variables remain.

Testing Significance of Canonical Correlations

For each step in the canonical correlation sequence, the significance of the smallest canonical correlation, ρ_{min}, is judged by an information criterion from Akaike (1976). This information criterion is

$$-n\ln(1-\rho_{min}^2) - \lambda(r(p+1)-q+1)$$

where *q* is the dimension of $\mathbf{f}_t^j$ at the current step, *r* is the order of the state vector, *p* is the order of the vector autoregressive process, and λ is the value of the SIGCORR= option. The default is SIGCORR=2. If this information criterion is less than or equal to 0, ρ_{min} is taken to be 0; otherwise, it is taken to be significantly greater than 0. (Do not confuse this information criterion with the AIC.)

Variables in $\mathbf{x}_{t+p|t}$ are not added in the model, even with positive information criterion, because of the singularity of $\mathbf{V}$. You can force the consideration of more candidate state variables by increasing the size of the $\mathbf{V}$ matrix by specifying a PASTMIN= option value larger than *p*.

Printing the Canonical Correlations

To print the details of the canonical correlation analysis process, specify the CANCORR option in the PROC STATESPACE statement. The CANCORR option prints the candidate state vectors, the canonical correlations, and the information criteria for testing the significance of the smallest canonical correlation.

Bartlett's χ^2 and its degrees of freedom are also printed when the CANCORR option is specified. The formula used for Bartlett's χ^2 is

$$\chi^2 = -(n - .5(r(p+1) - q + 1))\ln(1 - \rho^2_{min})$$

with $r(p+1) - q + 1$ degrees of freedom.

Figure 25.11 shows the output of the CANCORR option for the introductory example shown in the "Getting Started" section of this chapter.

```
                          The STATESPACE Procedure
                        Canonical Correlations Analysis

                                          Information        Chi
        x(T;T)       y(T;T)     x(T+1;T)    Criterion     Square       DF

             1            1     0.237045     3.566167    11.4505        4


                                                   Information        Chi
x(T;T)       y(T;T)     x(T+1;T)     y(T+1;T)        Criterion     Square       DF

     1            1     0.238244     0.056565         -5.35906   0.636134        3


                                                   Information        Chi
x(T;T)       y(T;T)     x(T+1;T)     x(T+2;T)        Criterion     Square       DF

     1            1     0.237602     0.087493         -4.46312   1.525353        3
```

Figure 25.11. Canonical Correlations Analysis

New variables are added to the state vector if the information criteria are positive. In this example, $\mathbf{y}_{t+1|t}$ and $\mathbf{x}_{t+2|t}$ are not added to the state space vector because the information criteria for these models are negative.

If the information criterion is nearly 0, then you may want to investigate models that arise if the opposite decision is made regarding ρ_{min}. This investigation can be accomplished by using a FORM statement to specify part or all of the state vector.

Preliminary Estimates of $\mathbf{F}$

When a candidate variable $x_{l,t+k|t}$ yields a zero ρ_{min} and is not added to the state vector, a linear combination of $\mathbf{f}^j_t$ is uncorrelated with the $\mathbf{p}_t$. Because of the method used to construct the $\mathbf{f}^j_t$ sequence, the coefficient of $x_{l,t+k|t}$ in $\mathbf{l}$ can be taken as 1. Denote the coefficients of $\mathbf{z}^j_t$ in this linear combination as $\mathbf{l}$.

This gives the relationship:

$$x_{l,t+k|t} = \mathbf{l}'\mathbf{z}_t^j$$

The vector $\mathbf{l}$ is used as a preliminary estimate of the first *r* columns of the row of the transition matrix $\mathbf{F}$ corresponding to $x_{l,t+k-1|t}$.

Parameter Estimation

The model is $\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$, where $\mathbf{e}_t$ is a sequence of independent multivariate normal innovations with mean vector **0** and variance $\Sigma_{\mathbf{ee}}$. The observed sequence, $\mathbf{x}_t$, composes the first *r* components of $\mathbf{z}_t$ and, thus, $\mathbf{x}_t = \mathbf{H}\mathbf{z}_t$, where $\mathbf{H}$ is the $r \times s$ matrix $[\mathbf{I}_r\ \mathbf{0}]$.

Let $\mathbf{E}$ be the $r \times n$ matrix of innovations:

$$\mathbf{E} = [\,\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_n\,]$$

If the number of observations, *n*, is reasonably large, the log likelihood, L, can be approximated up to an additive constant as follows:

$$L = -\frac{n}{2}\ln(|\boldsymbol{\Sigma}_{\mathbf{ee}}|) - \frac{1}{2}trace(\boldsymbol{\Sigma}_{\mathbf{ee}}^{-1}\mathbf{E}\mathbf{E}')$$

The elements of $\boldsymbol{\Sigma}_{\mathbf{ee}}$ are taken as free parameters and are estimated as follows:

$$\mathbf{S}_0 = \frac{1}{n}\mathbf{E}\mathbf{E}'$$

Replacing $\boldsymbol{\Sigma}_{\mathbf{ee}}$ by $\mathbf{S}_0$ in the likelihood equation, the log likelihood, up to an additive constant, is

$$\mathbf{L} = -\frac{n}{2}\ln(|\mathbf{S}_0|)$$

Letting B be the backshift operator, the formal relation between $\mathbf{x}_t$ and $\mathbf{e}_t$ is

$$\mathbf{x}_t = \mathbf{H}(\mathbf{I} - B\mathbf{F})^{-1}\mathbf{G}\mathbf{e}_t$$

$$\mathbf{e}_t = (\mathbf{H}(\mathbf{I} - B\mathbf{F})^{-1}\mathbf{G})^{-1}\mathbf{x}_t = \sum_{i=0}^{\infty} \boldsymbol{\Xi}_i \mathbf{x}_{t-i}$$

Letting $\mathbf{C}_i$ be the *i*th lagged sample covariance of $\mathbf{x}_t$, and neglecting end effects, the matrix $\mathbf{S}_0$ is

$$\mathbf{S}_0 = \sum_{i,j=0}^{\infty} \mathbf{\Xi}_i \mathbf{C}_{-i+j} \mathbf{\Xi}_j^{'}$$

For the computation of $\mathbf{S}_0$, the infinite sum is truncated at the value of the KLAG= option. The value of the KLAG= option should be large enough that the sequence $\mathbf{\Xi}_i$ is approximately 0 beyond that point.

Let θ be the vector of free parameters in the $\mathbf{F}$ and $\mathbf{G}$ matrices. The derivative of the log likelihood with respect to the parameter θ is

$$\frac{\partial L}{\partial \theta} = -\frac{n}{2} \,\mathrm{trace} \left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \theta} \right)$$

The second derivative is

$$\frac{\partial^2 \mathbf{L}}{\partial \theta \partial \theta'} = \frac{n}{2} \left(\mathrm{trace} \left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \theta'} \mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \theta} \right) - \mathrm{trace} \left(\mathbf{S}_0^{-1} \frac{\partial^2 \mathbf{S}_0}{\partial \theta \partial \theta'} \right) \right)$$

Near the maximum, the first term is unimportant and the second term can be approximated to give the following second derivative approximation:

$$\frac{\partial^2 L}{\partial \theta \partial \theta'} \cong -n \,\mathrm{trace} \left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{E}}{\partial \theta} \frac{\partial \mathbf{E}'}{\partial \theta'} \right)$$

The first derivative matrix and this second derivative matrix approximation are computed from the sample covariance matrix $\mathbf{C}_0$ and the truncated sequence $\mathbf{\Xi}_i$. The approximate likelihood function is maximized by a modified Newton-Raphson algorithm employing these derivative matrices.

The matrix $\mathbf{S}_0$ is used as the estimate of the innovation covariance matrix, $\mathbf{\Sigma_{ee}}$. The negative of the inverse of the second derivative matrix at the maximum is used as an approximate covariance matrix for the parameter estimates. The standard errors of the parameter estimates printed in the parameter estimates tables are taken from the diagonal of this covariance matrix. The parameter covariance matrix is printed when the COVB option is specified.

If the data are nearly nonstationary, a better estimate of $\mathbf{\Sigma_{ee}}$ and the other parameters can sometimes be obtained by specifying the RESIDEST option. The RESIDEST option estimates the parameters using conditional least squares instead of maximum likelihood.

The residuals are computed using the state space equation and the sample mean values of the variables in the model as start-up values. The estimate of $\mathbf{S}_0$ is then computed using the residuals from the *i*th observation on, where *i* is the maximum number of times any variable occurs in the state vector. A multivariate Gauss-Marquardt algorithm is used to minimize $|\mathbf{S}_0|$. Refer to Harvey (1981a) for a further description of this method.

Forecasting

Given estimates of $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{\Sigma_{ee}}$, forecasts of $\mathbf{x}_t$ are computed from the conditional expectation of $\mathbf{z}_t$.

In forecasting, the parameters $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{\Sigma_{ee}}$ are replaced with the estimates or by values specified in the RESTRICT statement. One-step-ahead forecasting is performed for the observation $\mathbf{x}_t$, where $t \leq n - b$. Here n is the number of observations and *b* is the value of the BACK= option. For the observation $\mathbf{x}_t$, where $t > n - b$, *m*-step-ahead forecasting is performed for $m = t - n + b$. The forecasts are generated recursively with the initial condition $\mathbf{z}_0 = 0$.

The *m*-step-ahead forecast of $\mathbf{z}_{t+m}$ is $\mathbf{z}_{t+m|t}$, where $\mathbf{z}_{t+m|t}$ denotes the conditional expectation of $\mathbf{z}_{t+m}$ given the information available at time *t*. The *m*-step-ahead forecast of $\mathbf{x}_{t+m}$ is $\mathbf{x}_{t+m|t} = \mathbf{H}\mathbf{z}_{t+m|t}$, where the matrix $\mathbf{H} = [\mathbf{I}_r \mathbf{0}]$.

Let $\Psi_i = \mathbf{F}^i\mathbf{G}$. Note that the last $s - r$ elements of $\mathbf{z}_t$ consist of the elements of $\mathbf{x}_{u|t}$ for $u > t$.

The state vector $\mathbf{z}_{t+m}$ can be represented as

$$\mathbf{z}_{t+m} = \mathbf{F}^m\mathbf{z}_t + \sum_{i=0}^{m-1} \mathbf{\Psi}_i\mathbf{e}_{t+m-i}$$

Since $\mathbf{e}_{t+i|t} = \mathbf{0}$ for $i > 0$, the *m*-step-ahead forecast $\mathbf{z}_{t+m|t}$ is

$$\mathbf{z}_{t+m|t} = \mathbf{F}^m\mathbf{z}_t = \mathbf{F}\mathbf{z}_{t+m-1|t}$$

Therefore, the *m*-step-ahead forecast of $\mathbf{x}_{t+m}$ is

$$\mathbf{x}_{t+m|t} = \mathbf{H}\mathbf{z}_{t+m|t}$$

The *m*-step-ahead forecast error is

$$\mathbf{z}_{t+m} - \mathbf{z}_{t+m|t} = \sum_{i=0}^{m-1} \mathbf{\Psi}_i\mathbf{e}_{t+m-i}$$

The variance of the *m*-step-ahead forecast error is

$$\mathbf{V}_{z,m} = \sum_{i=0}^{m-1} \boldsymbol{\Psi}_i \boldsymbol{\Sigma}_{\mathbf{ee}} \Psi_i'$$

Letting $\mathbf{V}_{z,0} = \mathbf{0}$, the variance of the *m*-step-ahead forecast error of $\mathbf{z}_{t+m}$, $\mathbf{V}_{z,m}$, can be computed recursively as follows:

$$\mathbf{V}_{z,m} = \mathbf{V}_{z,m-1} + \boldsymbol{\Psi}_{m-1} \boldsymbol{\Sigma}_{\mathbf{ee}} \boldsymbol{\Psi}_{m-1}^{'}$$

The variance of the *m*-step-ahead forecast error of $\mathbf{x}_{t+m}$ is the $r \times r$ left upper submatrix of $\mathbf{V}_{z,m}$; that is,

$$\mathbf{V}_{x,m} = \mathbf{H}\mathbf{V}_{z,m}\mathbf{H}'$$

Unless the NOCENTER option is specified, the sample mean vector is added to the forecast. When differencing is specified, the forecasts $\mathbf{x}_{t+m|t}$ plus the sample mean vector are integrated back to produce forecasts for the original series.

Let $\mathbf{y}_t$ be the original series specified by the VAR statement, with some 0 values appended corresponding to the unobserved past observations. Let B be the backshift operator, and let $\boldsymbol{\Delta}(B)$ be the $s \times s$ matrix polynomial in the backshift operator corresponding to the differencing specified by the VAR statement. The off-diagonal elements of $\boldsymbol{\Delta}_i$ are 0. Note that $\boldsymbol{\Delta}_0 = \mathbf{I}_s$, where $\mathbf{I}_s$ is the $s \times s$ identity matrix. Then $\mathbf{z}_t = \boldsymbol{\Delta}(B)\mathbf{y}_t$.

This gives the relationship

$$\mathbf{y}_t = \boldsymbol{\Delta}^{-1}(B)\mathbf{z}_t = \sum_{i=0}^{\infty} \boldsymbol{\Lambda}_i \mathbf{z}_{t-i}$$

where $\boldsymbol{\Delta}^{-1}(B) = \sum_{i=0}^{\infty} \boldsymbol{\Lambda}_i B^i$ and $\boldsymbol{\Lambda}_0 = \mathbf{I}_s$.

The *m*-step-ahead forecast of $\mathbf{y}_{t+m}$ is

$$\mathbf{y}_{t+m|t} = \sum_{i=0}^{m-1} \boldsymbol{\Lambda}_i \mathbf{z}_{t+m-i|t} + \sum_{i=m}^{\infty} \boldsymbol{\Lambda}_i \mathbf{z}_{t+m-i}$$

The *m*-step-ahead forecast error of $\mathbf{y}_{t+m}$ is

$$\sum_{i=0}^{m-1} \boldsymbol{\Lambda}_i \left(\mathbf{z}_{t+m-i} - \mathbf{z}_{t+m-i|t}\right) = \sum_{i=0}^{m-1} \left(\sum_{u=0}^{i} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{i-u}\right) \mathbf{e}_{t+m-i}$$

Letting $\mathbf{V}_{y,0} = \mathbf{0}$, the variance of the *m*-step-ahead forecast error of $\mathbf{y}_{t+m}$, $\mathbf{V}_{y,m}$, is

$$\begin{aligned}\mathbf{V}_{y,m} &= \sum_{i=0}^{m-1}\left(\sum_{u=0}^{i}\mathbf{\Lambda}_u\mathbf{\Psi}_{i-u}\right)\mathbf{\Sigma}_{\mathbf{ee}}\left(\sum_{u=0}^{i}\mathbf{\Lambda}_u\mathbf{\Psi}_{i-u}\right)' \\ &= \mathbf{V}_{y,m-1} + \left(\sum_{u=0}^{m-1}\mathbf{\Lambda}_u\mathbf{\Psi}_{m-1-u}\right)\mathbf{\Sigma}_{\mathbf{ee}}\left(\sum_{u=0}^{m-1}\mathbf{\Lambda}_u\mathbf{\Psi}_{m-1-u}\right)'\end{aligned}$$

Relation of ARMA and State Space Forms

Every state space model has an ARMA representation, and conversely every ARMA model has a state space representation. This section discusses this equivalence. The following material is adapted from Akaike (1974), where there is a more complete discussion. Pham-Dinh-Tuan (1978) also contains a discussion of this material.

Suppose you are given the following ARMA model:

$$\mathbf{\Phi}(B)\mathbf{x}_t = \mathbf{\Theta}(B)\mathbf{e}_t$$

or, in more detail

$$\mathbf{x}_t - \mathbf{\Phi}_1\mathbf{x}_{t-1} - \cdots - \mathbf{\Phi}_p\mathbf{x}_{t-p} = \mathbf{e}_t + \mathbf{\Theta}_1\mathbf{e}_{t-1} + \cdots + \mathbf{\Theta}_q\mathbf{e}_{t-q} \qquad (1)$$

where $\mathbf{e}_t$ is a sequence of independent multivariate normal random vectors with mean $\mathbf{0}$ and variance matrix $\mathbf{\Sigma}_{\mathbf{ee}}$; B is the backshift operator ($B\mathbf{x}_t = \mathbf{x}_{t-1}$); $\mathbf{\Phi}(B)$ and $\mathbf{\Theta}(B)$ are matrix polynomials in B; and $\mathbf{x}_t$ is the observed process.

If the roots of the determinantial equation $|\mathbf{\Phi}(B)| = 0$ are outside the unit circle in the complex plane, the model can also be written as

$$\mathbf{x}_t = \mathbf{\Phi}^{-1}(B)\mathbf{\Theta}(B)\mathbf{e}_t = \sum_{i=0}^{\infty}\mathbf{\Psi}_i\mathbf{e}_{t-i}$$

The $\mathbf{\Psi}_i$ matrices are known as the impulse response matrices and can be computed as $\mathbf{\Phi}^{-1}(B)\mathbf{\Theta}(B)$.

You can assume $p > q$ since, if this is not initially true, you can add more terms $\mathbf{\Phi}_i$ that are identically 0 without changing the model.

To write this set of equations in a state space form, proceed as follows. Let $\mathbf{x}_{t+i|t}$ be the conditional expectation of $\mathbf{x}_{t+i}$ given $\mathbf{x}_w$ for $w \leq t$. The following relations hold:

$$\mathbf{x}_{t+i|t} = \sum_{j=i}^{\infty}\mathbf{\Psi}_j\mathbf{e}_{t+i-j}$$

$$\mathbf{x}_{t+i|t+1} = \mathbf{x}_{t+i|t} + \mathbf{\Psi}_{i-1}\mathbf{e}_{t+1}$$

However, from equation (1) you can derive the following relationship:

$$\mathbf{x}_{t+p|t} = \mathbf{\Phi}_1\mathbf{x}_{t+p-1|t} + \cdots + \mathbf{\Phi}_p\mathbf{x}_t \qquad (2)$$

Hence, when $i = p$, you can substitute for $\mathbf{x}_{t+p|t}$ in the right-hand side of equation (2) and close the system of equations.

This substitution results in the following model in the state space form $\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{x}_{t+2|t+1} \\ \vdots \\ \mathbf{x}_{t+p|t+1} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{\Phi}_p & \mathbf{\Phi}_{p-1} & & \cdots & \mathbf{\Phi}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1|t} \\ \vdots \\ \mathbf{x}_{t+p-1|t} \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ \mathbf{\Psi}_1 \\ \vdots \\ \mathbf{\Psi}_{p-1} \end{bmatrix} \mathbf{e}_{t+1}$$

Note that the state vector $\mathbf{z}_t$ is composed of conditional expectations of $\mathbf{x}_t$ and the first *r* components of $\mathbf{z}_t$ are equal to $\mathbf{x}_t$.

The state space form can be cast into an ARMA form by solving the system of difference equations for the first *r* components.

When converting from an ARMA form to a state space form, you can generate a state vector larger than needed; that is, the state space model may not be a minimal representation. When going from a state space form to an ARMA form, you can have nontrivial common factors in the autoregressive and moving average operators that yield an ARMA model larger than necessary.

If the state space form used is not a minimal representation, some but not all components of $\mathbf{x}_{t+i|t}$ may be linearly dependent. This situation corresponds to $[\mathbf{\Phi}_p\mathbf{\Theta}_{p-1}]$ being of less than full rank when $\mathbf{\Phi}(B)$ and $\mathbf{\Theta}(B)$ have no common nontrivial left factors. In this case, $\mathbf{z}_t$ consists of a subset of the possible components of $[\mathbf{x}_{t+i|t}] \quad i = 1, 2, \cdots, p-1$. However, once a component of $\mathbf{x}_{t+i|t}$ (for example, the *j*th one) is linearly dependent on the previous conditional expectations, then all subsequent *j*th components of $\mathbf{x}_{t+k|t}$ for $k > i$ must also be linearly dependent. Note that in this case, equivalent but seemingly different structures can arise if the order of the components within $\mathbf{x}_t$ is changed.

OUT= Data Set

The forecasts are contained in the output data set specified by the OUT= option on the PROC STATESPACE statement. The OUT= data set contains the following variables:

- the BY variables
- the ID variable

- the VAR statement variables. These variables contain the actual values from the input data set.
- FOR*i*, numeric variables containing the forecasts. The variable FOR*i* contains the forecasts for the *i*th variable in the VAR statement list. Forecasts are one-step-ahead predictions until the end of the data or until the observation specified by the BACK= option.
- RES*i*, numeric variables containing the residual for the forecast of the *i*th variable in the VAR statement list. For forecast observations, the actual values are missing and the RES*i* variables contain missing values.
- STD*i*, numeric variables containing the standard deviation for the forecast of the *i*th variable in the VAR statement list. The values of the STD*i* variables can be used to construct univariate confidence limits for the corresponding forecasts. However, such confidence limits do not take into account the covariance of the forecasts.

OUTAR= Data Set

The OUTAR= data set contains the estimates of the preliminary autoregressive models. The OUTAR= data set contains the following variables:

- ORDER, a numeric variable containing the order p of the autoregressive model that the observation represents
- AIC, a numeric variable containing the value of the information criterion AIC_p
- SIGF*l*, numeric variables containing the estimate of the innovation covariance matrices for the forward autoregressive models. The variable SIGF*l* contains the *l*th column of $\hat{\mathbf{\Sigma}}_p$ in the observations with ORDER=*p*.
- SIGB*l*, numeric variables containing the estimate of the innovation covariance matrices for the backward autoregressive models. The variable SIGB*l* contains the *l*th column of $\hat{\mathbf{\Omega}}_p$ in the observations with ORDER=*p*.
- FOR*k_l*, numeric variables containing the estimates of the autoregressive parameter matrices for the forward models. The variable FOR*k_l* contains the *l*th column of the lag *k* autoregressive parameter matrix $\hat{\mathbf{\Phi}}^p_k$ in the observations with ORDER=*p*.
- BAC*k_l*, numeric variables containing the estimates of the autoregressive parameter matrices for the backward models. The variable BAC*k_l* contains the *l*th column of the lag *k* autoregressive parameter matrix $\hat{\mathbf{\Psi}}^p_k$ in the observations with ORDER=*p*.

The estimates for the order p autoregressive model can be selected as those observations with ORDER=*p*. Within these observations, the *k*,*l*th element of $\mathbf{\Phi}^p_i$ is given by the value of the FOR*i_l* variable in the *k*th observation. The *k*,*l*th element of $\mathbf{\Psi}^p_i$ is given by the value of BAC*i_l* variable in the *k*th observation. The *k*,*l*th element of $\mathbf{\Sigma}_p$ is given by SIGF*l* in the *k*th observation. The *k*,*l*th element of $\mathbf{\Omega}_p$ is given by SIGB*l* in the *k*th observation.

Table 25.1 shows an example of the OUTAR= data set, with ARMAX=3 and $\mathbf{x}_t$ of dimension 2. In Table 25.1, (i, j) indicate the *i*,*j*th element of the matrix.

Table 25.1. Values in the OUTAR= Data Set

Obs	ORDER	AIC	SIGF1	SIGF2	SIGB1	SIGB2	FOR1_1	FOR1_2	FOR2_1	FOR2_2	FOR3_1	FOR3_2	BACK1_1	BACK1_2	BACK2_1	BACK2_2	BACK3_1	BACK3_2
1	0	AIC_0	$\mathbf{\Sigma}_{0(1,1)}$	$\mathbf{\Sigma}_{0(1,2)}$	$\mathbf{\Omega}_{0(1,1)}$	$\mathbf{\Omega}_{0(1,2)}$	.	.	.	.	.	.	.	.	.	.	.	.
2	0	AIC_0	$\mathbf{\Sigma}_{0(2,1)}$	$\mathbf{\Sigma}_{0(2,2)}$	$\mathbf{\Omega}_{0(2,1)}$	$\mathbf{\Omega}_{0(2,2)}$	.	.	.	.	.	.	.	.	.	.	.	.
3	1	AIC_1	$\mathbf{\Sigma}_{1(1,1)}$	$\mathbf{\Sigma}_{1(1,2)}$	$\mathbf{\Omega}_{1(1,1)}$	$\mathbf{\Omega}_{1(1,2)}$	$\mathbf{\Phi}^1_{1(1,1)}$	$\mathbf{\Phi}^1_{1(1,2)}$	.	.	.	.	$\mathbf{\Psi}^1_{1(1,1)}$	$\mathbf{\Psi}^1_{1(1,2)}$	.	.	.	.
4	1	AIC_1	$\mathbf{\Sigma}_{1(2,1)}$	$\mathbf{\Sigma}_{1(1,2)}$	$\mathbf{\Omega}_{1(2,1)}$	$\mathbf{\Omega}_{1(1,2)}$	$\mathbf{\Phi}^1_{1(2,1)}$	$\mathbf{\Phi}^1_{1(2,2)}$	.	.	.	.	$\mathbf{\Psi}^1_{1(2,1)}$	$\mathbf{\Psi}^1_{1(2,2)}$	.	.	.	.
5	2	AIC_2	$\mathbf{\Sigma}_{2(1,1)}$	$\mathbf{\Sigma}_{2(1,2)}$	$\mathbf{\Omega}_{2(1,1)}$	$\mathbf{\Omega}_{2(1,2)}$	$\mathbf{\Phi}^2_{1(1,1)}$	$\mathbf{\Phi}^2_{1(1,2)}$	$\mathbf{\Phi}^2_{2(1,1)}$	$\mathbf{\Phi}^2_{2(1,2)}$	.	.	$\mathbf{\Psi}^2_{1(1,1)}$	$\mathbf{\Psi}^2_{1(1,2)}$	$\mathbf{\Psi}^2_{2(1,1)}$	$\mathbf{\Psi}^2_{2(1,2)}$	.	.
6	2	AIC_2	$\mathbf{\Sigma}_{2(2,1)}$	$\mathbf{\Sigma}_{2(1,2)}$	$\mathbf{\Omega}_{2(2,1)}$	$\mathbf{\Omega}_{2(1,2)}$	$\mathbf{\Phi}^2_{1(2,1)}$	$\mathbf{\Phi}^2_{1(2,2)}$	$\mathbf{\Phi}^2_{2(2,1)}$	$\mathbf{\Phi}^2_{2(2,2)}$	.	.	$\mathbf{\Psi}^2_{1(2,1)}$	$\mathbf{\Psi}^2_{1(2,2)}$	$\mathbf{\Psi}^2_{2(2,1)}$	$\mathbf{\Psi}^2_{2(2,2)}$	.	.
7	3	AIC_3	$\mathbf{\Sigma}_{3(1,1)}$	$\mathbf{\Sigma}_{3(1,2)}$	$\mathbf{\Omega}_{3(1,1)}$	$\mathbf{\Omega}_{3(1,2)}$	$\mathbf{\Phi}^3_{1(1,1)}$	$\mathbf{\Phi}^3_{1(1,2)}$	$\mathbf{\Phi}^3_{2(1,1)}$	$\mathbf{\Phi}^3_{2(1,2)}$	$\mathbf{\Phi}^3_{3(1,1)}$	$\mathbf{\Phi}^3_{3(1,2)}$	$\mathbf{\Psi}^3_{1(1,1)}$	$\mathbf{\Psi}^3_{1(1,2)}$	$\mathbf{\Psi}^3_{2(1,1)}$	$\mathbf{\Psi}^3_{2(1,2)}$	$\mathbf{\Psi}^3_{3(1,1)}$	$\mathbf{\Psi}^3_{3(1,2)}$
8	3	AIC_3	$\mathbf{\Sigma}_{3(2,1)}$	$\mathbf{\Sigma}_{3(1,2)}$	$\mathbf{\Omega}_{3(2,1)}$	$\mathbf{\Omega}_{3(1,2)}$	$\mathbf{\Phi}^3_{1(2,1)}$	$\mathbf{\Phi}^3_{1(2,2)}$	$\mathbf{\Phi}^3_{2(2,1)}$	$\mathbf{\Phi}^3_{2(2,2)}$	$\mathbf{\Phi}^3_{3(2,1)}$	$\mathbf{\Phi}^3_{3(2,2)}$	$\mathbf{\Psi}^3_{1(2,1)}$	$\mathbf{\Psi}^3_{1(2,2)}$	$\mathbf{\Psi}^3_{2(2,1)}$	$\mathbf{\Psi}^3_{2(2,2)}$	$\mathbf{\Psi}^3_{3(2,1)}$	$\mathbf{\Psi}^3_{3(2,2)}$

The estimated autoregressive parameters can be used in the IML procedure to obtain autoregressive estimates of the spectral density function or forecasts based on the autoregressive models.

OUTMODEL= Data Set

The OUTMODEL= data set contains the estimates of the $\mathbf{F}$ and $\mathbf{G}$ matrices and their standard errors, the names of the components of the state vector, and the estimates of the innovation covariance matrix. The variables contained in the OUTMODEL= data set are as follows:

- the BY variables
- STATEVEC, a character variable containing the name of the component of the state vector corresponding to the observation. The STATEVEC variable has the value STD for standard deviations observations, which contain the standard errors for the estimates given in the preceding observation.
- F_*j*, numeric variables containing the columns of the $\mathbf{F}$ matrix. The variable F_*j* contains the *j*th column of $\mathbf{F}$. The number of F_*j* variables is equal to the value of the DIMMAX= option. If the model is of smaller dimension, the extraneous variables are set to missing.
- G_*j*, numeric variables containing the columns of the $\mathbf{G}$ matrix. The variable G_*j* contains the *j*th column of $\mathbf{G}$. The number of G_*j* variables is equal to *r*, the dimension of $\mathbf{x}_t$ given by the number of variables in the VAR statement.
- SIG_*j*, numeric variables containing the columns of the innovation covariance matrix. The variable SIG_*j* contains the *j*th column of $\mathbf{\Sigma}_{ee}$. There are *r* variables SIG_*j*.

Table 25.2 shows an example of the OUTMODEL= data set, with $\mathbf{x}_t = (x_t, y_t)'$, $\mathbf{z}_t = (x_t, y_t, x_{t+1|t})'$, and DIMMAX=4. In Table 25.2, $\mathbf{F}_{i,j}$ and $\mathbf{G}_{i,j}$ are the *i*,*j*th elements of $\mathbf{F}$ and $\mathbf{G}$ respectively. Note that all elements for F_4 are missing because $\mathbf{F}$ is a 3×3 matrix.

Table 25.2. Value in the OUTMODEL= Data Set

Obs	STATEVEC	F_1	F_2	F_3	F_4	G_1	G_2	SIG_1	SIG_2
1	X(T;T)	0	0	1	.	1	0	$\mathbf{\Sigma}_{1,1}$	$\mathbf{\Sigma}_{1,2}$
2	STD	.	.	.	.	.	.	.	.
3	Y(T;T)	$\mathbf{F}_{2,1}$	$\mathbf{F}_{2,2}$	$\mathbf{F}_{2,3}$	.	0	1	$\mathbf{\Sigma}_{2,1}$	$\mathbf{\Sigma}_{2,2}$
4	STD	std $\mathbf{F}_{2,1}$	std $\mathbf{F}_{2,2}$	std $\mathbf{F}_{2,3}$	.	.	.	.	.
5	X(T+1;T)	$\mathbf{F}_{3,1}$	$\mathbf{F}_{3,2}$	$\mathbf{F}_{3,3}$	.	$\mathbf{G}_{3,1}$	$\mathbf{G}_{3,2}$	.	.
6	STD	std $\mathbf{F}_{3,1}$	std $\mathbf{F}_{3,2}$	std $\mathbf{F}_{3,3}$	.	std $\mathbf{G}_{3,1}$	std $\mathbf{G}_{3,2}$	.	.

Printed Output

The printed output produced by the STATESPACE procedure is described in the following:

1. descriptive statistics, which include the number of observations used, the names of the variables, their means and standard deviations (Std), and the differencing operations used.
2. the Akaike information criteria for the sequence of preliminary autoregressive models
3. if the PRINTOUT=LONG option is specified, the sample autocovariance matrices of the input series at various lags.
4. if the PRINTOUT=LONG option is specified, the sample autocorrelation matrices of the input series.
5. a schematic representation of the autocorrelation matrices, showing the significant autocorrelations.
6. if the PRINTOUT=LONG option is specified, the partial autoregressive matrices. (These are $\mathbf{\Phi}_p^p$ as described in "Preliminary Autoregressive Models" earlier in this chapter.)
7. a schematic representation of the partial autocorrelation matrices, showing the significant partial autocorrelations.
8. the Yule-Walker estimates of the autoregressive parameters for the autoregressive model with the minimum AIC.
9. if the PRINTOUT=LONG option is specified, the autocovariance matrices of the residuals of the minimum AIC model. This is the sequence of estimated innovation variance matrices for the solutions of the Yule-Walker equations.
10. if the PRINTOUT=LONG option is specified, the autocorrelation matrices of the residuals of the minimum AIC model.
11. If the CANCORR option is specified, the canonical correlations analysis for each potential state vector considered in the state vector selection process. This includes the potential state vector, the canonical correlations, the information criterion for the smallest canonical correlation, Bartlett's χ^2 statistic ("Chi Square") for the smallest canonical correlation, and the degrees of freedom of Bartlett's χ^2.
12. the components of the chosen state vector.
13. the preliminary estimate of the transition matrix, $\mathbf{F}$, the input matrix, $\mathbf{G}$, and the variance matrix for the innovations, $\mathbf{\Sigma_{ee}}$.

14. if the ITPRINT option is specified, the iteration history of the likelihood maximization. For each iteration, this shows the iteration number, the number of step halvings, the determinant of the innovation variance matrix, the damping factor Lambda, and the values of the parameters.
15. the state vector, printed again to aid interpretation of the following listing of $\mathbf{F}$ and $\mathbf{G}$.
16. the final estimate of the transition matrix, $\mathbf{F}$.
17. the final estimate of the input matrix, $\mathbf{G}$.
18. the final estimate of the variance matrix for the innovations, $\mathbf{\Sigma_{ee}}$.
19. a table listing the estimates of the free parameters in $\mathbf{F}$ and $\mathbf{G}$ and their standard errors and *t* statistics.
20. if the COVB option is specified, the covariance matrix of the parameter estimates.
21. if the COVB option is specified, the correlation matrix of the parameter estimates.
22. if the PRINT option is specified, the forecasts and their standard errors.

ODS Table Names

PROC STATESPACE assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 25.3. ODS Tables Produced in PROC STATESPACE

ODS Table Name	Description	Option
NObs	Number of observations	default
Summary	Simple summary statistics table	default
InfoCriterion	Information criterion table	default
CovLags	Covariance Matrices of Input Series	PRINTOUT=LONG
CorrLags	Correlation Matrices of Input Series	PRINTOUT=LONG
PartialAR	Partial Autoregressive Matrices	PRINTOUT=LONG
YWEstimates	Yule-Walker Estimates for Minimum AIC	default
CovResiduals	Covariance of Residuals	PRINTOUT=LONG
CorrResiduals	Residual Correlations from AR Models	PRINTOUT=LONG
StateVector	State vector table	default
CorrGraph	Schematic Representation of Correlations	default
TransitionMatrix	Transition Matrix	default
InputMatrix	Input Matrix	default
VarInnov	Variance Matrix for the Innovation	default
CovB	Covariance of Parameter Estimates	COVB
CorrB	Correlation of Parameter Estimates	COVB
CanCorr	Canonical Correlation Analysis	CANCORR
IterHistory	Iterative Fitting table	ITPRINT
ParameterEstimates	Parameter Estimates Table	default

Table 25.3. (continued)

ODS Table Name	Description	Option
Forecasts	Forecasts Table	PRINT
ConvergenceStatus	Convergence Status Table	default

Examples

Example 25.1. Series J from Box and Jenkins

This example analyzes the gas furnace data (series J) from Box and Jenkins. (The data are not shown. Refer to Box and Jenkins (1976) for the data.)

First, a model is selected and fit automatically using the following statements.

```
title1 'Gas Furnace Data';
title2 'Box & Jenkins Series J';
title3 'Automatically Selected Model';

proc statespace data=seriesj cancorr;
   var x y;
run;
```

The results for the automatically selected model are shown in Output 25.1.1.

Output 25.1.1. Results for Automatically Selected Model

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure

                       Number of Observations    296


                                             Standard
                      Variable        Mean       Error

                      x           -0.05683    1.072766
                      y           53.50912    3.202121



                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure

                 Information Criterion for Autoregressive Models

   Lag=0    Lag=1    Lag=2    Lag=3    Lag=4    Lag=5    Lag=6    Lag=7    Lag=8

651.3862 -1033.57 -1632.96 -1645.12 -1651.52 -1648.91 -1649.34 -1643.15 -1638.56

                                 Information
                                Criterion for
                               Autoregressive
                                   Models

                               Lag=9      Lag=10

                             -1634.8    -1633.59


                    Schematic Representation of Correlations

   Name/Lag     0     1     2     3     4     5     6     7     8     9    10

   x           +-    +-    +-    +-    +-    +-    +-    +-    +-    +-    +-
   y           -+    -+    -+    -+    -+    -+    -+    -+    -+    -+    -+

            + is > 2*std error,  - is < -2*std error,  . is between
```

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure

              Schematic Representation of Partial Autocorrelations

       Name/Lag     1     2     3     4     5     6     7     8     9    10

       x           +.    -.    +.    ..    ..    -.    ..    ..    ..    ..
       y           -+    --    -.    .+    ..    ..    ..    ..    ..    .+

             + is > 2*std error,  - is < -2*std error,  . is between


                     Yule-Walker Estimates for Minimum AIC

          ------Lag=1------ ------Lag=2------ ------Lag=3------ ------Lag=4------
                 x        y        x        y        x        y        x        y

x         1.925887 -0.00124 -1.20166 0.004224 0.116918 -0.00867 0.104236 0.003268
y         0.050496 1.299793 -0.02046  -0.3277 -0.71182 -0.25701 0.195411 0.133417
```

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure
                        Canonical Correlations Analysis

                                          Information        Chi
         x(T;T)      y(T;T)    x(T+1;T)     Criterion      Square      DF

              1           1    0.804883      292.9228    304.7481       8


                                                Information        Chi
   x(T;T)      y(T;T)    x(T+1;T)    y(T+1;T)     Criterion      Square      DF

        1           1    0.906681    0.607529      122.3358    134.7237       7


                                                   Information      Chi
    x(T;T)    y(T;T)  x(T+1;T)  y(T+1;T)  x(T+2;T)    Criterion    Square    DF

         1         1  0.909434  0.610278  0.186274     -1.54701  10.34705     6


                                                   Information      Chi
    x(T;T)    y(T;T)  x(T+1;T)  y(T+1;T)  y(T+2;T)    Criterion    Square    DF

         1         1   0.91014  0.618937  0.206823     0.940392  12.80924     6


                                                      Information      Chi
   x(T;T)   y(T;T) x(T+1;T) y(T+1;T) y(T+2;T) y(T+3;T)   Criterion   Square   DF

        1        1 0.912963 0.628785 0.226598 0.083258    -7.94103 2.041584    5
```

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                         Automatically Selected Model

                          The STATESPACE Procedure
                Selected Statespace Form and Preliminary Estimates

                               State Vector

  x(T;T)          y(T;T)          x(T+1;T)        y(T+1;T)        y(T+2;T)


                       Estimate of Transition Matrix

             0               0               1               0               0
             0               0               0               1               0
      -0.84718        0.026794        1.711715        -0.05019               0
             0               0               0               0               1
      -0.19785        0.334274        -0.18174        -1.23557        1.787475


                        Input Matrix for Innovation

                                     1               0
                                     0               1
                              1.925887        -0.00124
                              0.050496        1.299793
                              0.142421        1.361696
```

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                         Automatically Selected Model

                          The STATESPACE Procedure
                Selected Statespace Form and Preliminary Estimates

                       Variance Matrix for Innovation

                              0.035274        -0.00734
                              -0.00734        0.097569
                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure
                   Selected Statespace Form and Fitted Model

                                State Vector

     x(T;T)         y(T;T)         x(T+1;T)       y(T+1;T)       y(T+2;T)


                       Estimate of Transition Matrix

                0              0              1              0              0
                0              0              0              1              0
         -0.86192       0.030609       1.724235       -0.05483              0
                0              0              0              0              1
         -0.34839       0.292124       -0.09435       -1.09823       1.671418


                        Input Matrix for Innovation

                                      1              0
                                      0              1
                                1.92442       -0.00416
                               0.015621       1.258495
                                0.08058       1.353204
```

```
                              Gas Furnace Data
                           Box & Jenkins Series J
                        Automatically Selected Model

                          The STATESPACE Procedure
                   Selected Statespace Form and Fitted Model

                       Variance Matrix for Innovation

                               0.035579       -0.00728
                               -0.00728       0.095577


                            Parameter Estimates

                                             Standard
                   Parameter    Estimate        Error    t Value

                   F(3,1)       -0.86192     0.072961     -11.81
                   F(3,2)       0.030609     0.026167       1.17
                   F(3,3)       1.724235     0.061599      27.99
                   F(3,4)       -0.05483     0.030169      -1.82
                   F(5,1)       -0.34839     0.135253      -2.58
                   F(5,2)       0.292124     0.046299       6.31
                   F(5,3)       -0.09435     0.096527      -0.98
                   F(5,4)       -1.09823     0.109525     -10.03
                   F(5,5)       1.671418     0.083737      19.96
                   G(3,1)       1.924420     0.058162      33.09
                   G(3,2)       -0.00416     0.035255      -0.12
                   G(4,1)       0.015621     0.095771       0.16
                   G(4,2)       1.258495     0.055742      22.58
                   G(5,1)       0.080580     0.151622       0.53
                   G(5,2)       1.353204     0.091388      14.81
```

The two series are believed to have a transfer function relation with the gas rate (variable X) as the input and the CO_2 concentration (variable Y) as the output. Since the parameter estimates shown in Output 25.1.1 support this kind of model, the model is reestimated with the feedback parameters restricted to 0. The following statements fit the transfer function (no feedback) model.

```
title3 'Transfer Function Model';
proc statespace data=seriesj printout=none;
   var x y;
   restrict f(3,2)=0 f(3,4)=0
            g(3,2)=0 g(4,1)=0 g(5,1)=0;
run;
```

The last two pages of the output are shown in Output 25.1.2.

Output 25.1.2. STATESPACE Output for Transfer Function Model

```
                              Gas Furnace Data
                            Box & Jenkins Series J
                            Transfer Function Model

                          The STATESPACE Procedure
                     Selected Statespace Form and Fitted Model

                                 State Vector

   x(T;T)         y(T;T)         x(T+1;T)       y(T+1;T)       y(T+2;T)


                        Estimate of Transition Matrix

           0             0             1             0             0
           0             0             0             1             0
    -0.68882             0      1.598717             0             0
           0             0             0             0             1
    -0.35944      0.284179       -0.0963      -1.07313      1.650047


                         Input Matrix for Innovation

                                 1             0
                                 0             1
                          1.923446             0
                                 0      1.260856
                                 0      1.346332
                              Gas Furnace Data
                            Box & Jenkins Series J
                           Transfer Function Model

                          The STATESPACE Procedure
                    Selected Statespace Form and Fitted Model

                        Variance Matrix for Innovation

                           0.036995       -0.0072
                            -0.0072      0.095712


                             Parameter Estimates

                                              Standard
                   Parameter    Estimate         Error    t Value

                   F(3,1)       -0.68882    0.050549     -13.63
                   F(3,3)       1.598717    0.050924      31.39
                   F(5,1)       -0.35944    0.229044      -1.57
                   F(5,2)       0.284179    0.096944       2.93
                   F(5,3)       -0.09630    0.140876      -0.68
                   F(5,4)       -1.07313    0.250385      -4.29
                   F(5,5)       1.650047    0.188533       8.75
                   G(3,1)       1.923446    0.056328      34.15
                   G(4,2)       1.260856    0.056464      22.33
                   G(5,2)       1.346332    0.091086      14.78
```

References

Akaike, H. (1974), "Markovian Representation of Stochastic Processes and Its Application to the Analysis of Autoregressive Moving Average Processes," *Annals of the Institute of Statistical Mathematics*, 26, 363-387.

Akaike, H. (1976), "Canonical Correlations Analysis of Time Series and the Use of an Information Criterion," in *Advances and Case Studies in System Identification*, eds. R. Mehra and D.G. Lainiotis, New York: Academic Press.

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, New York: John Wiley & Sons.

Ansley, C.F. and Newbold, P. (1979), "Multivariate Partial Autocorrelations," *Proceedings of the Business and Economic Statistics Section*, American Statistical Association, 349-353.

Box, G.E.P. and Jenkins, G. (1976), *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.

Brockwell, P.J. and Davis, R.A. (1991), *Time Series: Theory and Methods*, 2nd Edition, Springer-Verlag.

Hannan, E.J. (1970), *Multiple Time Series*, New York: John Wiley & Sons.

Hannan, E.J. (1976), "The Identification and Parameterization of ARMAX and State Space Forms," *Econometrica*, 44, 713-722.

Harvey, A.C. (1981a), *The Econometric Analysis of Time Series*, New York: John Wiley & Sons.

Harvey, A.C. (1981b), *Time Series Models*, New York: John Wiley & Sons.

Jones, R.H. (1974), "Identification and Autoregressive Spectrum Estimation," *IEEE Transactions on Automatic Control*, AC-19, 894-897.

Pham-Dinh-Tuan (1978), "On the Fitting of Multivariate Processes of the Autoregressive-Moving Average Type," *Biometrika*, 65, 99-107.

Priestley, M.B. (1980), "System Identification, Kalman Filtering, and Stochastic Control," in *Directions in Time Series*, eds. D.R. Brillinger and G.C. Tiao, Institute of Mathematical Statistics.

Whittle, P. (1963), "On the Fitting of Multivariate Autoregressions and the Approximate Canonical Factorization of a Spectral Density Matrix," *Biometrika*, 50, 129-134.

Chapter 26
The SYSLIN Procedure

Chapter Contents

Chapter 26
The SYSLIN Procedure

Overview

The SYSLIN procedure estimates parameters in an interdependent system of linear regression equations.

Ordinary least squares (OLS) estimates are biased and inconsistent when current period endogenous variables appear as regressors in other equations in the system. The errors of a set of related regression equations are often correlated, and the efficiency of the estimates can be improved by taking these correlations into account. The SYSLIN procedure provides several techniques which produce consistent and asymptotically efficient estimates for systems of regression equations.

The SYSLIN procedure provides the following estimation methods:

- ordinary least squares (OLS)
- two-stage least squares (2SLS)
- limited information maximum likelihood (LIML)
- K-class
- seemingly unrelated regressions (SUR)
- iterated seemingly unrelated regressions (ITSUR)
- three-stage least squares (3SLS)
- iterated three-stage least squares (IT3SLS)
- full information maximum likelihood (FIML)
- minimum expected loss (MELO)

Other features of the SYSLIN procedure enable you to:

- impose linear restrictions on the parameter estimates.
- test linear hypotheses about the parameters.
- write predicted and residual values to an output SAS data set.
- write parameter estimates to an output SAS data set.
- write the crossproducts matrix (SSCP) to an output SAS data set.
- use raw data, correlations, covariances, or cross products as input.

Experimental graphics are now available with the SYSLIN procedure. For more information, see the "ODS Graphics" section on page 1514.

Getting Started

This section introduces the use of the SYSLIN procedure. The problem of dependent regressors is introduced using a supply-demand example. This section explains the terminology used for variables in a system of regression equations and introduces the SYSLIN procedure statements for declaring the roles the variables play. The syntax used for the different estimation methods and the output produced is shown.

An Example Model

In simultaneous systems of equations, endogenous variables are determined jointly rather than sequentially. Consider the following demand and supply functions for some product:

$$Q_D = a_1 + b_1 P + c_1 Y + d_1 S + \epsilon_1 \qquad \text{(demand)}$$

$$Q_S = a_2 + b_2 P + c_2 U + \epsilon_2 \qquad \text{(supply)}$$

$$Q = Q_D = Q_S \qquad \text{(market equilibrium)}$$

The variables in this system are as follows:

Q$_D$	quantity demanded
Q$_S$	quantity supplied
Q	the observed quantity sold, which equates quantity supplied and quantity demanded in equilibrium
P	price per unit
Y	income
S	price of substitutes
U	unit cost
ϵ_1	the random error term for the demand equation
ϵ_2	the random error term for the supply equation

In this system, quantity demanded depends on price, income, and the price of substitutes. Consumers normally purchase more of a product when prices are lower and when income and the price of substitute goods are higher. Quantity supplied depends on price and the unit cost of production. Producers will supply more when price is high and when unit cost is low. The actual price and quantity sold are determined jointly by the values that equate demand and supply.

Since price and quantity are jointly endogenous variables, both structural equations are necessary to adequately describe the observed values. A critical assumption of OLS is that the regressors are uncorrelated with the residual. When current endogenous variables appear as regressors in other equations (endogenous variables depend on each other), this assumption is violated and the OLS parameter estimates are biased and inconsistent. The bias caused by the violated assumptions is called *Simultaneous equation bias*. Neither the demand nor supply equation can be estimated consistently by OLS.

Variables in a System of Equations

Before explaining how to use the SYSLIN procedure, it is useful to define some terms. The variables in a system of equations can be classified as follows:

- *Endogenous variables*, which are also called *jointly dependent* or *response variables*, are the variables determined by the system. Endogenous variables can also appear on the right-hand side of equations.
- *Exogenous variables* are independent variables that do not depend on any of the endogenous variables in the system.
- *Predetermined variables* include both the exogenous variables and *lagged endogenous variables*, which are past values of endogenous variables determined at previous time periods. PROC SYSLIN does not compute lagged values; any lagged endogenous variables must be computed in a preceding DATA step.
- *Instrumental variables* are predetermined variables used in obtaining predicted values for the current period endogenous variables by a first-stage regression. The use of instrumental variables characterizes estimation methods such as two-stage least squares and three-stage least squares. Instrumental variables estimation methods substitute these first-stage predicted values for endogenous variables when they appear as regressors in model equations.

Using PROC SYSLIN

First specify the input data set and estimation method on the PROC SYSLIN statement. If any model uses dependent regressors, and you are using an instrumental variables regression method, declare the dependent regressors with an ENDOGENOUS statement and declare the instruments with an INSTRUMENTS statement. Next, use MODEL statements to specify the structural equations of the system.

The use of different estimation methods is shown by the following examples. These examples use the simulated dataset WORK.IN given below.

```
data in;
  label q = \quotes{Quantity}
        p = \quotes{Price}
        s = \quotes{Price of Substitutes}
        y = \quotes{Income}
        u = \quotes{Unit Cost};
  drop i e1 e2;
  p = 0; q = 0;
  do i = 1 to 60;
     y = 1 + .05*i  + .15*rannor(123);
     u = 2          + .05*rannor(123) + .05*rannor(123);
```

```
          s = 4 - .001*(i-10)*(i-110) + .5*rannor(123);
          e1 = .15 * rannor(123);
          e2 = .15 * rannor(123);
          demandx = 1 + .3 * y + .35 * s + e1;
          supplyx = -1 - 1 * u + e2 - .4*e1;
          q = 1.4/2.15 * demandx + .75/2.15 * supplyx;
          p = ( - q + supplyx ) / -1.4;
          output;
       end;
    run;
```

OLS Estimation

PROC SYSLIN performs OLS regression if you do not specify a method of estimation in the PROC SYSLIN statement. OLS does not use instruments, so the ENDOGENOUS and INSTRUMENTS statements can be omitted.

The following statements estimate the supply and demand model shown previously:

```
proc syslin data=in;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The PROC SYSLIN output for the demand equation is shown in Figure 26.1, and the output for the supply equation is shown in Figure 26.2.

```
                              The SYSLIN Procedure
                         Ordinary Least Squares Estimation

                           Model                   DEMAND
                           Dependent Variable           q
                           Label                 Quantity


                              Analysis of Variance

                                     Sum of        Mean
     Source                 DF      Squares      Square    F Value    Pr > F

     Model                   3     9.587891    3.195964     398.31    <.0001
     Error                  56     0.449336    0.008024
     Corrected Total        59     10.03723


             Root MSE              0.08958    R-Square       0.95523
             Dependent Mean        1.30095    Adj R-Sq       0.95283
             Coeff Var             6.88541


                              Parameter Estimates

                 Parameter Standard                   Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.47677 0.210239   -2.27   0.0272 Intercept
p              1  0.123324 0.105177    1.17   0.2459 Price
y              1  0.201282 0.032403    6.21   <.0001 Income
s              1  0.167258 0.024091    6.94   <.0001 Price of Substitutes
```

Figure 26.1. OLS Results for Demand Equation

```
                          The SYSLIN Procedure
                     Ordinary Least Squares Estimation

                      Model                     SUPPLY
                      Dependent Variable             q
                      Label                   Quantity


                            Analysis of Variance

                                      Sum of        Mean
     Source                 DF       Squares      Square    F Value    Pr > F

     Model                   2      9.033890    4.516945     256.61    <.0001
     Error                  57      1.003337    0.017602
     Corrected Total        59      10.03723


             Root MSE             0.13267    R-Square       0.90004
             Dependent Mean       1.30095    Adj R-Sq       0.89653
             Coeff Var           10.19821


                            Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.30390 0.471397   -0.64   0.5217 Intercept
p              1  1.218743 0.053914   22.61   <.0001 Price
u              1  -1.07757 0.234150   -4.60   <.0001 Unit Cost
```

Figure 26.2. OLS Results for Supply Equation

For each MODEL statement, the output first shows the model label and dependent variable name and label. This is followed by an Analysis of Variance table for the model, which shows the model, error, and total mean squares, and an *F* test for the no-regression hypothesis. Next, the procedure prints the root mean square error, dependent variable mean and coefficient of variation, and the R^2 and adjusted R^2 statistics.

Finally, the table of parameter estimates shows the estimated regression coefficients, standard errors, and *t*-tests. You would expect the price coefficient in a demand equation to be negative. However, note that the OLS estimate of the price coefficient P in the demand equation (.1233) has a positive sign. This could be caused by simultaneous equation bias.

Two-Stage Least Squares Estimation

In the supply and demand model, P is an endogenous variable, and consequently the OLS estimates are biased. The following example estimates this model using two-stage least squares.

```
proc syslin data=in 2sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The 2SLS option on the PROC SYSLIN statement specifies the two-stage least-squares method. The ENDOGENOUS statement specifies that P is an endogenous regressor for which first-stage predicted values are substituted. You only need to declare an endogenous variable in the ENDOGENOUS statement if it is used as a regressor; thus although Q is endogenous in this model, it is not necessary to list it in the ENDOGENOUS statement.

Usually, all predetermined variables that appear in the system are used as instruments. The INSTRUMENTS statement specifies that the exogenous variables Y, U, and S are used as instruments for the first-stage regression to predict P.

The 2SLS results are shown in Figure 26.3 and Figure 26.4. The first-stage regressions are not shown. To see the first-stage regression results, use the FIRST option on the MODEL statement.

```
                             The SYSLIN Procedure
                      Two-Stage Least Squares Estimation

                        Model                    DEMAND
                        Dependent Variable            q
                        Label                  Quantity


                             Analysis of Variance

                                     Sum of        Mean
     Source                  DF     Squares      Square    F Value    Pr > F

     Model                    3    9.670882    3.223627     115.58    <.0001
     Error                   56    1.561944    0.027892
     Corrected Total         59    10.03723


             Root MSE            0.16701    R-Square       0.86095
             Dependent Mean      1.30095    Adj R-Sq       0.85350
             Coeff Var          12.83740


                             Parameter Estimates

                Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  1.901040 1.171224    1.62   0.1102 Intercept
p              1  -1.11518 0.607391   -1.84   0.0717 Price
y              1  0.419544 0.117954    3.56   0.0008 Income
s              1  0.331475 0.088472    3.75   0.0004 Price of Substitutes
```

Figure 26.3. 2SLS Results for Demand Equation

```
                              The SYSLIN Procedure
                       Two-Stage Least Squares Estimation

                         Model                    SUPPLY
                         Dependent Variable            q
                         Label                  Quantity


                              Analysis of Variance

                                    Sum of        Mean
     Source                 DF     Squares      Square    F Value    Pr > F

     Model                   2    9.646098    4.823049     253.96    <.0001
     Error                  57    1.082503    0.018991
     Corrected Total        59    10.03723


             Root MSE             0.13781    R-Square       0.89910
             Dependent Mean       1.30095    Adj R-Sq       0.89556
             Coeff Var           10.59291


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.51878 0.490999   -1.06   0.2952 Intercept
p              1  1.333080 0.059271   22.49   <.0001 Price
u              1  -1.14623 0.243491   -4.71   <.0001 Unit Cost
```

Figure 26.4. 2SLS Results for Supply Equation

The 2SLS output is similar in form to the OLS output. However, the 2SLS results are based on predicted values for the endogenous regressors from the first stage instrumental regressions. This makes the analysis of variance table and the R^2 statistics difficult to interpret. See the sections "ANOVA Table for Instrumental Variables Methods" and "The R^2 Statistics" later in this chapter for details.

Note that, unlike the OLS results, the 2SLS estimate for the P coefficient in the demand equation (-1.115) is negative.

LIML, K-Class, and MELO Estimation

To obtain limited information maximum likelihood, general K-class, or minimum expected loss estimates, use the ENDOGENOUS, INSTRUMENTS, and MODEL statements as in the 2SLS case but specify the LIML, K=, or MELO option instead of 2SLS in the PROC SYSLIN statement. The following statements show this for K-class estimation.

```
proc syslin data=in k=.5;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

For more information on these estimation methods see the “Estimation Methods” in the “Details” section and consult econometrics textbooks.

SUR, 3SLS, and FIML Estimation

In a multivariate regression model, the errors in different equations may be correlated. In this case the efficiency of the estimation may be improved by taking these cross-equation correlations into account.

Seemingly Unrelated Regression

Seemingly unrelated regression (SUR), also called joint generalized least squares (JGLS) or Zellner estimation, is a generalization of OLS for multi-equation systems. Like OLS, the SUR method assumes that all the regressors are independent variables, but SUR uses the correlations among the errors in different equations to improve the regression estimates. The SUR method requires an initial OLS regression to compute residuals. The OLS residuals are used to estimate the cross-equation covariance matrix.

The SUR option on the PROC SYSLIN statement specifies seemingly unrelated regression, as shown in the following statements:

```
proc syslin data=in sur;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

INSTRUMENTS and ENDOGENOUS statements are not needed for SUR, since the SUR method assumes there are no endogenous regressors. For SUR to be effective, the models must use different regressors. SUR produces the same results as OLS unless the model contains at least one regressor not used in the other equations.

Three-Stage Least Squares

The three-stage least-squares method generalizes the two-stage least-squares method to take account of the correlations between equations in the same way that SUR generalizes OLS. Three-stage least squares requires three steps: first-stage regressions to get predicted values for the endogenous regressors; a two-stage least-squares step to get residuals to estimate the cross-equation correlation matrix; and the final 3SLS estimation step.

The 3SLS option on the PROC SYSLIN statement specifies the three-stage least-squares method, as shown in the following statements.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The 3SLS output begins with a two-stage least-squares regression to estimate the cross-model correlation matrix. This output is the same as the 2SLS results shown in Figure 26.3 and Figure 26.4, and is not repeated here. The next part of the 3SLS output prints the cross-model correlation matrix computed from the 2SLS residuals. This output is shown in Figure 26.5 and includes the cross-model covariances, correlations, the inverse of the correlation matrix, and the inverse covariance matrix.

```
                         The SYSLIN Procedure
                 Three-Stage Least Squares Estimation

                        Cross Model Covariance

                          DEMAND        SUPPLY

                 DEMAND        0.027892      -.011283
                 SUPPLY        -.011283      0.018991


                       Cross Model Correlation

                          DEMAND        SUPPLY

                 DEMAND         1.00000      -0.49022
                 SUPPLY        -0.49022       1.00000


                   Cross Model Inverse Correlation

                          DEMAND        SUPPLY

                 DEMAND         1.31634       0.64530
                 SUPPLY         0.64530       1.31634


                    Cross Model Inverse Covariance

                          DEMAND        SUPPLY

                 DEMAND         47.1945       28.0380
                 SUPPLY         28.0380       69.3130
```

Figure 26.5. Estimated Cross-Model Covariances used for 3SLS Estimates

The final 3SLS estimates are shown in Figure 26.6.

```
                              The SYSLIN Procedure
                      Three-Stage Least Squares Estimation

                     System Weighted MSE            0.5711
                     Degrees of freedom                113
                     System Weighted R-Square       0.9627


                         Model                  DEMAND
                         Dependent Variable          q
                         Label                Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  1.980261 1.169169    1.69   0.0959 Intercept
p              1  -1.17654 0.605012   -1.94   0.0568 Price
y              1  0.404115 0.117179    3.45   0.0011 Income
s              1  0.359204 0.085077    4.22   <.0001 Price of Substitutes


                         Model                  SUPPLY
                         Dependent Variable          q
                         Label                Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.51878 0.490999   -1.06   0.2952 Intercept
p              1  1.333080 0.059271   22.49   <.0001 Price
u              1  -1.14623 0.243491   -4.71   <.0001 Unit Cost
```

Figure 26.6. Three-Stage Least Squares Results

This output first prints the system weighted mean square error and system weighted R^2 statistics. The system weighted MSE and system weighted R^2 measure the fit of the joint model obtained by stacking all the models together and performing a single regression with the stacked observations weighted by the inverse of the model error variances. See the section "The R^2 Statistics" for details.

Next, the table of 3SLS parameter estimates for each model is printed. This output has the same form as for the other estimation methods.

Note that the 3SLS and 2SLS results may be the same in some cases. This results from the same principle that causes OLS and SUR results to be identical unless an equation includes a regressor not used in the other equations of the system. However, the application of this principle is more complex when instrumental variables are used. When all the exogenous variables are used as instruments, linear combinations of all the exogenous variables appear in the third-stage regressions through substitution of first-stage predicted values.

In this example, 3SLS produces different (and, it is hoped, more efficient) estimates for the demand equation. However, the 3SLS and 2SLS results for the supply equation are the same. This is because the supply equation has one endogenous regressor and one exogenous regressor not used in other equations. In contrast, the demand equation has fewer endogenous regressors than exogenous regressors not used in other equations in the system.

Full Information Maximum Likelihood

The FIML option on the PROC SYSLIN statement specifies the full information maximum likelihood method, as shown in the following statements.

```
proc syslin data=in fiml;
   endogenous  p q;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The FIML results are shown in Figure 26.7.

```
                          The SYSLIN Procedure
              Full-Information Maximum Likelihood Estimation

        NOTE: Convergence criterion met at iteration 3.


                         Model                   DEMAND
                         Dependent Variable           q
                         Label                 Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  1.988529 1.233625    1.61   0.1126 Intercept
p              1  -1.18147 0.652274   -1.81   0.0755 Price
y              1  0.402310 0.107269    3.75   0.0004 Income
s              1  0.361345 0.103816    3.48   0.0010 Price of Substitutes


                         Model                   SUPPLY
                         Dependent Variable           q
                         Label                 Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.52443 0.479522   -1.09   0.2787 Intercept
p              1  1.336083 0.057939   23.06   <.0001 Price
u              1  -1.14804 0.237793   -4.83   <.0001 Unit Cost
```

Figure 26.7. FIML Results

Computing Reduced-Form Estimates

A system of structural equations with endogenous regressors can be represented as functions only of the predetermined variables. For this to be possible, there must be as many equations as endogenous variables. If there are more endogenous variables than regression models, you can use IDENTITY statements to complete the system. See "Reduced-Form Estimates" in the "Computational Details" section later in this chapter for details.

The REDUCED option on the PROC SYSLIN statement prints reduced form estimates. The following statements show this using the 3SLS estimates of the structural parameters.

```
proc syslin data=in 3sls reduced;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The first four pages of this output were as shown previously and are not repeated here. (See Figure 26.3, Figure 26.4, Figure 26.5, and Figure 26.6.) The final page of the output from this example contains the reduced-form coefficients from the 3SLS structural estimates, as shown in Figure 26.8.

```
                               The SYSLIN Procedure
                        Three-Stage Least Squares Estimation

                               Endogenous Variables

                                            p             q

                       DEMAND        1.176539             1
                       SUPPLY        -1.33308             1


                               Exogenous Variables

                      Intercept             y             s             u

        DEMAND         1.980261      0.404115      0.359204             0
        SUPPLY         -0.51878             0             0      -1.14623


                          Inverse Endogenous Variables

                                  DEMAND        SUPPLY

                          p      0.398467      -0.39847
                          q      0.531188      0.468812


                                  Reduced Form

                   Intercept             y             s             u

            p       0.995786      0.161027      0.143131      0.456736
            q        0.80868      0.214661      0.190805      -0.53737
```

Figure 26.8. Reduced-Form 3SLS Results

Restricting Parameter Estimates

You can impose restrictions on the parameter estimates with RESTRICT and SRESTRICT statements. The RESTRICT statement imposes linear restrictions on parameters in the equation specified by the preceding MODEL statement. The SRESTRICT statement imposes linear restrictions that relate parameters in different models.

To impose restrictions involving parameters in different equations, use the SRESTRICT statement. Specify the parameters in the linear hypothesis as *model-label.regressor-name*. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.)

Tests for the significance of the restrictions are printed when RESTRICT or SRESTRICT statements are used. You can label RESTRICT and SRESTRICT statements to identify the restrictions in the output.

The RESTRICT statement in the following example restricts the price coefficient in the demand equation to equal .015. The SRESTRICT statement restricts the estimate of the income coefficient in the demand equation to be .01 times the estimate of the unit cost coefficient in the supply equation.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   peq015: restrict p = .015;
   supply: model q = p u;
   yeq01u: srestrict demand.y = .01 * supply.u;
run;
```

The restricted estimation results are shown in Figure 26.9.

```
                              The SYSLIN Procedure
                       Three-Stage Least Squares Estimation

                          Model                   DEMAND
                          Dependent Variable           q
                          Label                 Quantity


                               Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.46584 0.053307   -8.74   <.0001 Intercept
p              1  0.015000        0     .       .    Price
y              1  -0.00679 0.002357   -2.88   0.0056 Income
s              1  0.325589 0.009872   32.98   <.0001 Price of Substitutes
RESTRICT      -1  50.59341 7.464990    6.78   <.0001 PEQ015


                          Model                   SUPPLY
                          Dependent Variable           q
                          Label                 Quantity


                               Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -1.31894 0.477633   -2.76   0.0077 Intercept
p              1  1.291718 0.059101   21.86   <.0001 Price
u              1  -0.67887 0.235679   -2.88   0.0056 Unit Cost


                               Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

RESTRICT      -1  342.3611 38.12103    8.98   <.0001 YEQ01U
```

Figure 26.9. Restricted Estimates

The standard error for P in the demand equation is 0, since the value of the P coefficient was specified by the RESTRICT statement and not estimated from the data. The Parameter Estimates table for the demand equation contains an additional row for the restriction specified by the RESTRICT statement. The “parameter estimate” for the restriction is the value of the Lagrange multiplier used to impose the restriction.

The restriction is highly “significant” ($t = 6.777$), which means that the data are not consistent with the restriction, and the model does not fit as well with the restriction imposed. See the section “RESTRICT Statement” for more information.

After the Parameter Estimates table for the supply equation, the results for the cross model restrictions are printed. This shows that the restriction specified by the SRESTRICT statement is not consistent with the data ($t = 8.98$). See the section “SRESTRICT Statement” for more information.

Testing Parameters

You can test linear hypotheses about the model parameters with TEST and STEST statements. The TEST statement tests hypotheses about parameters in the equation specified by the preceding MODEL statement. The STEST statement tests hypotheses that relate parameters in different models.

For example, the following statements test the hypothesis that the price coefficient in the demand equation is equal to .015.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   test_1: test p = .015;
   supply: model q = p u;
run;
```

The TEST statement results are shown in Figure 26.10. This reports an *F*-test for the hypothesis specified by the TEST statement. In this case the *F* statistic is 6.79 (3.879/.571) with 1 and 113 degrees of freedom. The *p*-value for this *F* statistic is .0104, which indicates that the hypothesis tested is almost but not quite rejected at the .01 level. See the section “TEST Statement” for more information.

```
                              The SYSLIN Procedure
                       Three-Stage Least Squares Estimation

                      System Weighted MSE            0.5711
                      Degrees of freedom                113
                      System Weighted R-Square       0.9627


                          Model                   DEMAND
                          Dependent Variable           q
                          Label                 Quantity


                               Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  1.980261 1.169169    1.69   0.0959 Intercept
p              1  -1.17654 0.605012   -1.94   0.0568 Price
y              1  0.404115 0.117179    3.45   0.0011 Income
s              1  0.359204 0.085077    4.22   <.0001 Price of Substitutes


                        Test Results for Variable TEST_1

                      Num DF      Den DF    F Value    Pr > F

                           1         113       6.79    0.0104
```

Figure 26.10. TEST Statement Results

To test hypotheses involving parameters in different equations, use the STEST statement. Specify the parameters in the linear hypothesis as *model-label.regressor-name*. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.)

For example, the following statements test the hypothesis that the income coefficient in the demand equation is .01 times the unit cost coefficient in the supply equation:

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
   stest1: stest demand.y = .01 * supply.u;
run;
```

The STEST statement results are shown in Figure 26.11. The form and interpretation of the STEST statement results is like the TEST statement results. In this case, the *F*-test produces a *p*-value less than .0001, and strongly rejects the hypothesis tested. See the section “STEST Statement” for more information.

```
                              The SYSLIN Procedure
                       Three-Stage Least Squares Estimation

                      System Weighted MSE           0.5711
                      Degrees of freedom               113
                      System Weighted R-Square      0.9627


                          Model                  DEMAND
                          Dependent Variable          q
                          Label                Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  1.980261 1.169169    1.69   0.0959 Intercept
p              1  -1.17654 0.605012   -1.94   0.0568 Price
y              1  0.404115 0.117179    3.45   0.0011 Income
s              1  0.359204 0.085077    4.22   <.0001 Price of Substitutes


                          Model                  SUPPLY
                          Dependent Variable          q
                          Label                Quantity


                              Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.51878 0.490999   -1.06   0.2952 Intercept
p              1  1.333080 0.059271   22.49   <.0001 Price
u              1  -1.14623 0.243491   -4.71   <.0001 Unit Cost


                        Test Results for Variable STEST1

                      Num DF      Den DF     F Value    Pr > F

                           1         113       22.46    0.0001
```

Figure 26.11. STEST Statement Results

You can combine TEST and STEST statements with RESTRICT and SRESTRICT statements to perform hypothesis tests for restricted models. Of course, the validity of the TEST and STEST statement results will depend on the correctness of any restrictions you impose on the estimates.

Saving Residuals and Predicted Values

You can store predicted values and residuals from the estimated models in a SAS data set. Specify the OUT= option on the PROC SYSLIN statement and use the OUTPUT statement to specify names for new variables to contain the predicted and residual values.

For example, the following statements store the predicted quantity from the supply and demand equations in a data set PRED:

```
proc syslin data=in out=pred 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   output predicted=q_demand;
   supply: model q = p u;
   output predicted=q_supply;
run;
```

Plotting Residuals

You can plot the residuals against the regressors by specifying the PLOT option on the MODEL statement. For example, the following statements plot the 2SLS residuals for the demand model against price, income, price of substitutes, and the intercept.

```
proc syslin data=in 2sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s / plot;
run;
```

The plot for price is shown in Figure 26.12. The other plots are not shown.

```
         --+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+--
  DEMAND |                                                                     |



                              The SYSLIN Procedure
                       Two-Stage Least Squares Estimation

         |                                                                     |
         |                                                                     |
     0.6 +                                                                     +
         |                                                                     |
Q        |                                                                     |
u        |                                                                     |
a    0.4 +                                                                     +
n        |                                               1 1                   |
t        |                                                                     |
i        |                                       111                           |
t    0.2 +                                  2      1 1                         +
y        |                    1    1  11              1 1                      |
         |                   1   1  1                  1                       |
         |                     2  1       1    11 1       11                   |
     0.0 +                           1    11      2                            +
         |              1  1   1            1    1       1                     |
         |                 1       1    1  1  1  1   1 1                       |
         |                  1 1    1        1 1                                |
    -0.2 +                            1         2       1                      +
         |                       1                                             |
         |                   1           1                                     |
         |                                  1                                  |
    -0.4 +                                                                     +
         |                                                                     |
         |                                                                     |
         |                                                                     |
    -0.6 +                                                                     +
         |                                                                     |
         |                                                                     |
         --+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+--
          2.0   2.2   2.4   2.6   2.8   3.0   3.2   3.4   3.6   3.8   4.0   4.2

                            Price                  p
```

Figure 26.12. PLOT Option Output for P

Syntax

The SYSLIN procedure uses the following statements:

PROC SYSLIN *options* **;**
BY *variables* **;**
ENDOGENOUS *variables* **;**
IDENTITY *identities* **;**
INSTRUMENTS *variables* **;**
MODEL *response = regressors / options* **;**
OUTPUT PREDICTED= *variable* **RESIDUAL=** *variable* **;**
RESTRICT *restrictions* **;**
SRESTRICT *restrictions* **;**
STEST *equations* **;**
TEST *equations* **;**
VAR *variables* **;**
WEIGHT *variable* **;**

Functional Summary

The SYSLIN procedure statements and options are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set	PROC SYSLIN	DATA=
specify the output data set	PROC SYSLIN	OUT=
write parameter estimates to an output data set	PROC SYSLIN	OUTEST=
write covariances to the OUTEST= data set	PROC SYSLIN	OUTCOV
		OUTCOV3
write the SSCP matrix to an output data set	PROC SYSLIN	OUTSSCP=
Estimation Method Options		
specify full information maximum likelihood estimation	PROC SYSLIN	FIML
specify iterative SUR estimation	PROC SYSLIN	ITSUR
specify iterative 3SLS estimation	PROC SYSLIN	IT3SLS
specify K-class estimation	PROC SYSLIN	K=
specify limited information maximum likelihood estimation	PROC SYSLIN	LIML
specify minimum expected loss estimation	PROC SYSLIN	MELO
specify ordinary least squares estimation	PROC SYSLIN	OLS
specify seemingly unrelated estimation	PROC SYSLIN	SUR

Description	Statement	Option
specify two-stage least-squares estimation	PROC SYSLIN	2SLS
specify three-stage least-squares estimation	PROC SYSLIN	3SLS
specify Fuller's modification to LIML	PROC SYSLIN	ALPHA=
specify convergence criterion	PROC SYSLIN	CONVERGE=
specify maximum number of iterations	PROC SYSLIN	MAXIT=
use diagonal of **S** instead of **S**	PROC SYSLIN	SDIAG
exclude RESTRICT statements in final stage	PROC SYSLIN	NOINCLUDE
specify criterion for testing for singularity	PROC SYSLIN	SINGULAR=
specify denominator for variance estimates	PROC SYSLIN	VARDEF=
Printing Control Options		
print first-stage regression statistics	PROC SYSLIN	FIRST
print estimates and SSE at each iteration	PROC SYSLIN	ITPRINT
print the restricted reduced-form estimates	PROC SYSLIN	REDUCED
print descriptive statistics	PROC SYSLIN	SIMPLE
print uncorrected SSCP matrix	PROC SYSLIN	USSCP
print correlations of the parameter estimates	MODEL	CORRB
print covariances of the parameter estimates	MODEL	COVB
print Durbin-Watson statistics	MODEL	DW
print Basmann's test	MODEL	OVERID
plot residual values against regressors	MODEL	PLOT
print standardized parameter estimates	MODEL	STB
print unrestricted parameter estimates	MODEL	UNREST
print the model crossproducts matrix	MODEL	XPX
print the inverse of the crossproducts matrix	MODEL	I
suppress printed output	MODEL	NOPRINT
suppress all printed output	PROC SYSLIN	NOPRINT
Model Specification		
specify structural equations	MODEL	
suppress the intercept parameter	MODEL	NOINT
specify linear relationship among variables	IDENTITY	
perform weighted regression	WEIGHT	
Tests and Restrictions on Parameters		
place restrictions on parameter estimates	RESTRICT	
place restrictions on parameter estimates	SRESTRICT	
test linear hypothesis	STEST	
test linear hypothesis	TEST	
Other Statements		

Description	**Statement**	**Option**
specify BY-group processing	BY	
specify the endogenous variables	ENDOGENOUS	
specify instrumental variables	INSTRUMENTS	
write predicted and residual values to a data set	OUTPUT	
name variable for predicted values	OUTPUT	PREDICTED=
name variable for residual values	OUTPUT	RESIDUAL=
include additional variables in $X'X$ matrix	VAR	

PROC SYSLIN Statement

PROC SYSLIN *options;*

The following options can be used with the PROC SYSLIN statement.

Data Set Options

DATA= *SAS-data-set*
specifies the input data set. If the DATA= option is omitted, the most recently created SAS data set is used. In addition to ordinary SAS data sets, PROC SYSLIN can analyze data sets of TYPE=CORR, TYPE=COV, TYPE=UCORR, TYPE=UCOV, and TYPE=SSCP. See “Special TYPE= Input Data Set” in the “Input Data Set” section later in this chapter for more information.

OUT= *SAS-data-set*
specifies an output SAS data set for residuals and predicted values. The OUT= option is used in conjunction with the OUTPUT statement. See the section “OUT= Data Set” later in this chapter for more details.

OUTEST= *SAS-data-set*
writes the parameter estimates to an output data set. See the section “OUTEST= Data Set” later in this chapter for details.

OUTCOV
COVOUT
writes the covariance matrix of the parameter estimates to the OUTEST= data set in addition to the parameter estimates.

OUTCOV3
COV3OUT
writes covariance matrices for each model in a system to the OUTEST= data set when the 3SLS, SUR, or FIML option is used.

OUTSSCP= *SAS-data-set*
writes the sum-of-squares-and-crossproducts matrix to an output data set. See the section “OUTSSCP= Data Set” later in this chapter for details.

Estimation Method Options

2SLS
specifies the two-stage least-squares estimation method.

3SLS
specifies the three-stage least-squares estimation method.

FIML
specifies the full information maximum likelihood estimation method.

ITSUR
specifies the iterative seemingly unrelated estimation method.

IT3SLS
specifies the iterative three-stage least-squares estimation method.

K= *value*
specifies the K-class estimation method.

LIML
specifies the limited information maximum likelihood estimation method.

MELO
specifies the minimum expected loss estimation method.

OLS
specifies the ordinary least squares estimation method. This is the default.

SUR
specifies the seemingly unrelated estimation method.

Printing and Control Options

ALL
specifies the CORRB, COVB, DW, I, OVERID, PLOT, STB, and XPX options for every MODEL statement.

ALPHA= *value*
specifies Fuller’s modification to the LIML estimation method. See “Fuller’s Modification to LIML *K* Value” later in this chapter for details.

CONVERGE= *value*
specifies the convergence criterion for the iterative estimation methods IT3SLS, ITSUR, and FIML. The default is CONVERGE=.0001.

FIRST
prints first-stage regression statistics for the endogenous variables regressed on the instruments. This output includes sums of squares, estimates, variances, and standard deviations.

ITPRINT

prints parameter estimates, system-weighted residual sum of squares, and R^2 at each iteration for the IT3SLS and ITSUR estimation methods. For the FIML method, the ITPRINT option prints parameter estimates, negative of log likelihood function, and norm of gradient vector at each iteration.

MAXITER= *n*

specifies the maximum number of iterations allowed for the IT3SLS, ITSUR, and FIML estimation methods. The MAXITER= option can be abbreviated as MAXIT=. The default is MAXITER=30.

NOINCLUDE

excludes the RESTRICT statements from the final stage for the 3SLS, IT3SLS, SUR, ITSUR estimation methods.

NOPRINT

suppresses all printed output. Specifying NOPRINT in the PROC SYSLIN statement is equivalent to specifying NOPRINT in every MODEL statement.

REDUCED

prints the reduced-form estimates. If the REDUCED option is specified, you should specify any IDENTITY statements needed to make the system square. See “Reduced-Form Estimates” in the section “Computational Details” later in this chapter for more information.

SDIAG

uses the diagonal of **S** instead of **S** to do the estimation, where **S** is the covariance matrix of equation errors. See “Uncorrelated Errors Across Equations” in the section “Computational Details” later in this chapter for more information.

SIMPLE

prints descriptive statistics for the dependent variables. The statistics printed include the sum, mean, uncorrected sum of squares, variance, and standard deviation.

SINGULAR= *value*

specifies a criterion for testing singularity of the crossproducts matrix. This is a tuning parameter used to make PROC SYSLIN more or less sensitive to singularities. The value must be between 0 and 1. The default is SINGULAR=1E-8.

USSCP

prints the uncorrected sum-of-squares-and-crossproducts matrix.

USSCP2

prints the uncorrected sum-of-squares-and-crossproducts matrix for all variables used in the analysis, including predicted values of variables generated by the procedure.

VARDEF= DF | N | WEIGHT | WGT

specifies the denominator to use in calculating cross-equation error covariances and parameter standard errors and covariances. The default is VARDEF=DF, which corrects for model degrees of freedom. VARDEF=N specifies no degrees-of-freedom correction. VARDEF=WEIGHT specifies the sum of the observation weights. VARDEF=WGT specifies the sum of the observation weights minus the model degrees of freedom. See "Computation of Standard Errors" in the section "Computational Details" later in this chapter for more information.

BY Statement

BY *variables ;*

A BY statement can be used with PROC SYSLIN to obtain separate analyses on observations in groups defined by the BY variables.

ENDOGENOUS Statement

ENDOGENOUS *variables ;*

The ENDOGENOUS statement declares the jointly dependent variables that are projected in the first-stage regression through the instrument variables. The ENDOGENOUS statement is not needed for the SUR, ITSUR, or OLS estimation methods. The default ENDOGENOUS list consists of all the dependent variables in the MODEL and IDENTITY statements that do not appear in the INSTRUMENTS statement.

IDENTITY Statement

IDENTITY *equation ;*

The IDENTITY statement specifies linear relationships among variables to write to the OUTEST= data set. It provides extra information in the OUTEST= data set but does not create or compute variables. The OUTEST= data set can be processed by the SIMLIN procedure in a later step.

The IDENTITY statement is also used to compute reduced-form coefficients when the REDUCED option in the PROC SYSLIN statement is specified. See "Reduced-Form Estimates" in the section "Computational Details" later in this chapter for more information.

The *equation* given by the IDENTITY statement has the same form as equations in the MODEL statement. A label can be specified for an IDENTITY statement as follows:

label: **IDENTITY** . . . **;**

INSTRUMENTS Statement

INSTRUMENTS *variables ;*

The INSTRUMENTS statement declares the variables used in obtaining first-stage predicted values. All the instruments specified are used in each first-stage regression. The INSTRUMENTS statement is required for the 2SLS, 3SLS, IT3SLS, LIML, MELO, and K-class estimation methods. The INSTRUMENTS statement is not needed for the SUR, ITSUR, OLS, or FIML estimation methods.

MODEL Statement

MODEL *response = regressors / options ;*

The MODEL statement regresses the response variable on the left side of the equal sign against the regressors listed on the right side.

Models can be given labels. Model labels are used in the printed output to identify the results for different models. Model labels are also used in SRESTRICT and STEST statements to refer to parameters in different models. If no label is specified, the response variable name is used as the label for the model. The model label is specified as follows:

label: **MODEL** . . . **;**

The following options can be used in the MODEL statement after a slash (/).

ALL
specifies the CORRB, COVB, DW, I, OVERID, PLOT, STB, and XPX options.

ALPHA= *value*
specifies the α parameter for Fuller's modification to the LIML estimation method. See "Fuller's Modification to LIML" in the section "Computational Details" later in this chapter for more information.

CORRB
prints the matrix of estimated correlations between the parameter estimates.

COVB
prints the matrix of estimated covariances between the parameter estimates.

DW
prints Durbin-Watson statistics and autocorrelation coefficients for the residuals. If there are missing values, d' is calculated according to Savin and White (1978). Use the DW option only if the data set to be analyzed is an ordinary SAS data set with time series observations sorted in time order. The Durbin-Watson test is not valid for models with lagged dependent regressors.

I
prints the inverse of the crossproducts matrix for the model, $(\mathbf{X}'\mathbf{X})^{-1}$. If restrictions are specified, the crossproducts matrix printed is adjusted for the restrictions. See the section "Computational Details" for more information.

K= *value*
specifies K-class estimation.

NOINT
suppresses the intercept parameter from the model.

NOPRINT
suppresses the normal printed output.

OVERID
prints Basmann's (1960) test for over identifying restrictions. See "Over Identification Restrictions" in the section "Computational Details" later in this chapter for more information.

PLOT
plots residual values against regressors. A plot of the residuals for each regressor is printed.

STB
prints standardized parameter estimates. Sometimes known as a standard partial regression coefficient, a standardized parameter estimate is a parameter estimate multiplied by the standard deviation of the associated regressor and divided by the standard deviation of the response variable.

UNREST
prints parameter estimates computed before restrictions are applied. The UNREST option is valid only if a RESTRICT statement is specified.

XPX
prints the model crossproducts matrix, $X'X$. See the section "Computational Details" for more information.

OUTPUT Statement

OUTPUT PREDICTED=*variable* **RESIDUAL=***variable* **;**

The OUTPUT statement writes predicted values and residuals from the preceding model to the data set specified by the OUT= option on the PROC SYSLIN statement. An OUTPUT statement must come after the MODEL statement to which it applies. The OUT= option must be specified in the PROC SYSLIN statement.

The following options can be specified in the OUTPUT statement:

PREDICTED= *variable*
names a new variable to contain the predicted values for the response variable. The PREDICTED= option can be abbreviated as PREDICT=, PRED=, or P=.

RESIDUAL= *variable*
names a new variable to contain the residual values for the response variable. The RESIDUAL= option can be abbreviated as RESID= or R=.

For example, the following statements create an output data set named B. In addition to the variables in the input data set, the data set B contains the variable YHAT, with values that are predicted values of the response variable Y, and YRESID, with values that are the residual values of Y.

```
proc syslin data=a out=b;
   model y = x1 x2;
   output p=yhat r=yresid;
run;
```

For example, the following statements create an output data set named PRED. In addition to the variables in the input data set, the data set PRED contains the variables Q_DEMAND and Q_SUPPLY, with values that are predicted values of the response variable Q for the demand and supply equations respectively, and R_DEMAND and R_SUPPLY, with values that are the residual values of the demand and supply equations.

```
proc syslin data=in out=pred;
   demand: model q = p y s;
   output p=q_demand r=r_demand;
   supply: model q = p u;
   output p=q_supply r=r_supply;
run;
```

See the section “OUT= Data Set” later in this chapter for more details.

RESTRICT Statement

RESTRICT *equation , ... , equation ;*

The RESTRICT statement places restrictions on the parameter estimates for the preceding MODEL statement. Any number of restrict statements can follow a MODEL statement. Each restriction is written as a linear equation. If more than one restriction is specified in a single RESTRICT statement, the restrictions are separated by commas.

Parameters are referred to by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. The keyword INTERCEPT is used to refer to the intercept parameter in the model.

RESTRICT statements can be given labels. The labels are used in the printed output to distinguish results for different restrictions. Labels are specified as follows:

label **: RESTRICT** . . . **;**

The following is an example of the use of the RESTRICT statement, in which the coefficients of the regressors X1 and X2 are required to sum to 1.

```
proc syslin data=a;
   model y = x1 x2;
   restrict x1 + x2 = 1;
run;
```

Variable names can be multiplied by constants. When no equal sign appears, the linear combination is set equal to 0. Note that the parameters associated with the variables are restricted, not the variables themselves. Here are some examples of valid RESTRICT statements:

```
restrict x1 + x2 = 1;
restrict x1 + x2 - 1;
restrict 2 * x1 = x2 + x3 , intercept + x4 = 0;
restrict x1 = x2 = x3 = 1;
restrict 2 * x1 - x2;
```

Restricted parameter estimates are computed by introducing a Lagrangian parameter λ for each restriction (Pringle and Raynor 1971). The estimates of these Lagrangian parameters are printed in the parameter estimates table. If a restriction cannot be applied, its parameter value and degrees of freedom are listed as 0.

The Lagrangian parameter, λ, measures the sensitivity of the SSE to the restriction. If the restriction is changed by a small amount ϵ, the SSE is changed by $2\lambda\epsilon$.

The *t*-ratio tests the significance of the restrictions. If λ is zero, the restricted estimates are the same as the unrestricted.

Any number of restrictions can be specified on a RESTRICT statement, and any number of RESTRICT statements can be used. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

Note: The RESTRICT statement is not supported for the FIML estimation method.

SRESTRICT Statement

SRESTRICT *equation , ... , equation ;*

The SRESTRICT statement imposes linear restrictions involving parameters in two or more MODEL statements. The SRESTRICT statement is like the RESTRICT statement but is used to impose restrictions across equations, whereas the RESTRICT statement only applies to parameters in the immediately preceding MODEL statement.

Each restriction is written as a linear equation. Parameters are referred to as *label.variable*, where *label* is the model label and *variable* is the name of the regressor to which the parameter is attached. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.) Each variable name used must be a regressor in the indicated MODEL statement. The keyword INTERCEPT is used to refer to intercept parameters.

SRESTRICT statements can be given labels. The labels are used in the printed output to distinguish results for different restrictions. Labels are specified as follows:

label **: SRESTRICT** . . . **;**

The following is an example of the use of the SRESTRICT statement, in which the coefficient for the regressor X2 is constrained to be the same in both models.

```
proc syslin data=a 3sls;
   endogenous y1 y2;
   instruments x1 x2;
   model y1 = y2 x1 x2;
   model y2 = y1 x2;
   srestrict y1.x2 = y2.x2;
run;
```

When no equal sign is used, the linear combination is set equal to 0. Thus the restriction in the preceding example can also be specified as

```
   srestrict y1.x2 - y2.x2;
```

Any number of restrictions can be specified on an SRESTRICT statement, and any number of SRESTRICT statements can be used. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

When a system restriction is requested for a single equation estimation method (such as OLS or 2SLS), PROC SYSLIN produces the restricted estimates by actually using a corresponding system method. For example, when SRESTRICT is specified along with OLS, PROC SYSLIN produces the restricted OLS estimates via a two-step process equivalent to using SUR estimation with the SDIAG option. First of all, the unrestricted OLS results are produced. Then the GLS (SUR) estimation with the system restriction is performed using the diagonal of the covariance matrix of the residuals. When SRESTRICT is specified along with 2SLS, PROC SYSLIN produces the restricted 2SLS estimates via a multistep process equivalent to using 3SLS estimation with the SDIAG option. First of all, the unrestricted 2SLS results are produced. Then the GLS (3SLS) estimation with the system restriction is performed using the diagonal of the covariance matrix of the residuals.

The results of the SRESTRICT statements are printed after the parameter estimates for all the models in the system. The format of the SRESTRICT statement output is the same as the parameter estimates table. In this output the "Parameter Estimate" is the Lagrangian parameter, λ, used to impose the restriction.

The Lagrangian parameter, λ, measures the sensitivity of the system sum of square errors to the restriction. The system SSE is the system MSE shown in the printed output multiplied by the degrees of freedom. If the restriction is changed by a small amount ϵ, the system SSE is changed by $2\lambda\epsilon$.

The *t*-ratio tests the significance of the restriction. If λ is zero, the restricted estimates are the same as the unrestricted estimates.

The model degrees of freedom are not adjusted for the cross-model restrictions imposed by SRESTRICT statements.

Note: The SRESTRICT statement is not supported for the LIML and the FIML estimation methods.

STEST Statement

STEST *equation , ... , equation / options ;*

The STEST statement performs an *F*-test for the joint hypotheses specified in the statement.

The hypothesis is represented in matrix notation as

$$\mathbf{L}\beta = \mathbf{c}$$

and the *F*-test is computed as

$$\frac{(\mathbf{L}b - \mathbf{c})'(\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}')^{-1}(\mathbf{L}b - \mathbf{c})}{m\hat{\sigma}^2}$$

where *b* is the estimate of β, *m* is the number of restrictions, and $\hat{\sigma}^2$ is the system weighted mean square error. See the section "Computational Details" for information on the matrix $\mathbf{X}'\mathbf{X}$.

Each hypothesis to be tested is written as a linear equation. Parameters are referred to as *label.variable*, where *label* is the model label and *variable* is the name of the regressor to which the parameter is attached. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.) Each variable name used must be a regressor in the indicated MODEL statement. The keyword INTERCEPT is used to refer to intercept parameters.

STEST statements can be given labels. The label is used in the printed output to distinguish different tests. Any number of STEST statements can be specified. Labels are specified as follows:

label: **STEST** . . . **;**

The following is an example of the STEST statement:

```
proc syslin data=a 3sls;
   endogenous y1 y2;
   instruments x1 x2;
   model y1 = y2 x1 x2;
   model y2 = y1 x2;
   stest y1.x2 = y2.x2;
run;
```

The test performed is exact only for ordinary least squares, given the OLS assumptions of the linear model. For other estimation methods, the *F*-test is based on large sample theory and is only approximate in finite samples.

If RESTRICT or SRESTRICT statements are used, the tests computed by the STEST statement are conditional on the restrictions specified. The validity of the tests may be compromised if incorrect restrictions are imposed on the estimates.

The following are examples of STEST statements:

```
stest a.x1 + b.x2 = l;
stest 2 * b.x2 = c.x3 + c.x4 ,
      a.intercept + b.x2 = 0;
stest a.x1 = c.x2 = b.x3 = 1;
stest 2 * a.x1 - b.x2 = 0;
```

The PRINT option can be specified in the STEST statement after a slash (/):

PRINT
prints intermediate calculations for the hypothesis tests.

Note: The STEST statement is not supported for the FIML estimation method.

TEST Statement

TEST *equation , ... , equation / options ;*

The TEST statement performs *F*-tests of linear hypotheses about the parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. If more than one equation is specified, the equations are separated by commas.

Variable names must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT is used to refer to the model intercept.

TEST statements can be given labels. The label is used in the printed output to distinguish different tests. Any number of TEST statements can be specified. Labels are specified as follows:

label: **TEST** . . . **;**

The following is an example of the use of TEST statement, which tests the hypothesis that the coefficients of X1 and X2 are the same:

```
proc syslin data=a;
   model y = x1 x2;
   test x1 = x2;
run;
```

The following statements perform *F*-tests for the hypothesis that the coefficients of X1 and X2 are equal, and that the sum of the X1 and X2 coefficients is twice the intercept, and for the joint hypothesis.

```
proc syslin data=a;
   model y = x1 x2;
   x1eqx2:  test x1 = x2;
   sumeq2i: test x1 + x2 = 2 * intercept;
   joint:   test x1 = x2, x1 + x2 = 2 * intercept;
run;
```

The following are additional examples of TEST statements:

```
test x1 + x2 = 1;
test x1 = x2 = x3 = 1;
test 2 * x1 = x2 + x3, intercept + x4 = 0;
test 2 * x1 - x2;
```

The TEST statement performs an *F*-test for the joint hypotheses specified. The hypothesis is represented in matrix notation as follows:

$$\mathbf{L}\beta = \mathbf{c}$$

The *F* test is computed as

$$\frac{(\mathbf{L}b - \mathbf{c})'(\mathbf{L}(\mathbf{X}'\mathbf{X})^{-}\mathbf{L}')^{-1}(\mathbf{L}b - \mathbf{c})}{m\hat{\sigma}^2}$$

where *b* is the estimate of β, *m* is the number of restrictions, and $\hat{\sigma}^2$ is the model mean square error. See the section "Computational Details" for information on the matrix $\mathbf{X}'\mathbf{X}$.

The test performed is exact only for ordinary least squares, given the OLS assumptions of the linear model. For other estimation methods, the *F*-test is based on large sample theory and is only approximate in finite samples.

If RESTRICT or SRESTRICT statements are used, the tests computed by the TEST statement are conditional on the restrictions specified. The validity of the tests may be compromised if incorrect restrictions are imposed on the estimates.

The PRINT option can be specified in the TEST statement after a slash (/):

PRINT
prints intermediate calculations for the hypothesis tests.

Note: The TEST statement is not supported for the FIML estimation method.

VAR Statement

VAR *variables ;*

The VAR statement is used to include variables in the crossproducts matrix that are not specified in any MODEL statement. This statement is rarely used with PROC SYSLIN and is used only with the OUTSSCP= option in the PROC SYSLIN statement.

WEIGHT Statement

WEIGHT *variable ;*

The WEIGHT statement is used to perform weighted regression. The WEIGHT statement names a variable in the input data set whose values are relative weights for a weighted least-squares fit. If the weight value is proportional to the reciprocal of the variance for each observation, the weighted estimates are the best linear unbiased estimates (BLUE).

Details

Input Data Set

PROC SYSLIN does not compute new values for regressors. For example, if you need a lagged variable, you must create it with a DATA step. No values are computed by IDENTITY statements; all values must be in the input data set.

Special TYPE= Input Data Set

The input data set for most applications of the SYSLIN procedure contains standard rectangular data. However, PROC SYSLIN can also process input data in the form of a crossproducts, covariance, or correlation matrix. Data sets containing such matrices are identified by values of the TYPE= data set option.

These special kinds of input data sets can be used to save computer time. It takes nk^2 operations, where n is the number of observations and k is the number of variables, to calculate cross products; the regressions are of the order k^3. When n is in the thousands and k is much smaller, you can save most of the computer time in later runs of PROC SYSLIN by reusing the SSCP matrix rather than recomputing it.

The SYSLIN procedure can process TYPE= CORR, COV, UCORR, UCOV, or SSCP data sets. TYPE=CORR and TYPE=COV data sets, usually created by the CORR procedure, contain means and standard deviations, and correlations or covariances. TYPE=SSCP data sets, usually created in previous runs of PROC SYSLIN, contain sums of squares and cross products. Refer to *SAS/STAT User's Guide* for more information on special SAS data sets.

When special SAS data sets are read, you must specify the TYPE= data set option. PROC CORR and PROC SYSLIN automatically set the type for output data sets; however, if you create the data set by some other means, you must specify its type with the TYPE= data set option.

When the special data sets are used, the DW (Durbin-Watson test) and PLOT options in the MODEL statement cannot be performed, and the OUTPUT statements are not valid.

Estimation Methods

A brief description of the methods used by the SYSLIN procedure follows. For more information on these methods, see the references at the end of this chapter.

There are two fundamental methods of estimation for simultaneous equations: least squares and maximum likelihood. There are two approaches within each of these categories: single equation methods, also referred to as limited information methods, and system methods, or full information methods. System methods take into account cross-equation correlations of the disturbances in estimating parameters, while single equation methods do not.

OLS, 2SLS, MELO, K-class, SUR, ITSUR, 3SLS, and IT3SLS use the least-squares method; LIML and FIML use the maximum likelihood method.

OLS, 2SLS, MELO, K-class and LIML are single equation methods. The system methods are SUR, ITSUR, 3SLS, IT3SLS, and FIML.

Single Equation Estimation Methods

Single equation methods do not take into account correlations of errors across equations. As a result, these estimators are not asymptotically efficient compared to full information methods, however, there are instances in which they may be preferred. (See "Choosing a Method for Simultaneous Equations" later in this chapter for more information.)

Let $\mathbf{y}_i$ be the dependent endogenous variable in equation i, and X_i and Y_i be the matrices of exogenous and endogenous variables appearing as regressors in the same equation.

The 2SLS method owes its name to the fact that, in a first stage, the instrumental variables are used as regressors to obtain a projected value $\hat{Y}_i$ that is uncorrelated with the residual in equation i. In a second stage, $\hat{Y}_i$ replaces Y_i on the right hand side to obtain consistent least squares estimators.

Normally, the predetermined variables of the system are used as the instruments. It is possible to use variables other than predetermined variables from your system as instruments, however, the estimation may not be as efficient. For consistent estimates, the instruments must be uncorrelated with the residual and correlated with the endogenous variables.

The LIML method results in consistent estimates that are equal to the 2SLS estimates when an equation is exactly identified. LIML can be viewed as a least-variance ratio estimation or as a maximum likelihood estimation. LIML involves minimizing the ratio $\lambda = (rvar\_eq)/(rvar\_sys)$, where $rvar\_eq$ is the residual variance associated with regressing the weighted endogenous variables on all predetermined variables appearing in that equation, and $rvar\_sys$ is the residual variance associated with regressing weighted endogenous variables on all predetermined variables in the system.

The MELO method computes the minimum expected loss estimator. MELO estimators “minimize the posterior expectation of generalized quadratic loss functions for structural coefficients of linear structural models” (Judge et al. 1985, p. 635).

K-class estimators are a class of estimators that depends on a user-specified parameter k. A *K*-value less than 1 is recommended but not required. k may be deterministic or stochastic, but its probability limit must equal 1 for consistent parameter estimates. When all the predetermined variables are listed as instruments, they include all the other single equation estimators supported by PROC SYSLIN. The instance when some of the predetermined variables are not listed among the instruments is not supported by PROC SYSLIN for the general K-class estimation. It is, however, supported for the other methods.

For $k = 1$, the K-class estimator is the 2SLS estimator, while for $k = 0$, the K-class estimator is the OLS estimator. The K-class interpretation of LIML is that $k = \lambda$. Note that *k* is stochastic in the LIML method, unlike for OLS and 2SLS.

MELO is a Bayesian K-class estimator. It yields estimates that can be expressed as a matrix-weighted average of the OLS and 2SLS estimates. MELO estimators have finite second moments and hence finite risk. Other frequently used K-class estimators may not have finite moments under some commonly encountered circumstances and hence there can be infinite risk relative to quadratic and other loss functions.

One way of comparing K-class estimators is to note that when *k*=1, the correlation between regressor and the residual is completely corrected for. In all other cases, it is only partially corrected for.

See “Computational Details” later in this section for more details on K-class estimators.

SUR and 3SLS Estimation Methods

SUR may improve the efficiency of parameter estimates when there is contemporaneous correlation of errors across equations. In practice, the contemporaneous correlation matrix is estimated using OLS residuals. Under two sets of circumstances, SUR parameter estimates are the same as those produced by OLS: when there is no contemporaneous correlation of errors across equations (the estimate of contemporaneous correlation matrix is diagonal,) and when the independent variables are the same across equations.

Theoretically, SUR parameter estimates will always be at least as efficient as OLS in large samples, provided that your equations are correctly specified. However, in small samples the need to estimate the covariance matrix from the OLS residuals increases the sampling variability of the SUR estimates, and this effect can cause SUR to be less efficient than OLS. If the sample size is small and the across-equation correlations are small, then OLS should be preferred to SUR. The consequences of specification error are also more serious with SUR than with OLS.

The 3SLS method combines the ideas of the 2SLS and SUR methods. Like 2SLS, the 3SLS method uses $\hat{Y}$ instead of Y for endogenous regressors, which results in consistent estimates. Like SUR, the 3SLS method takes the cross-equation error correlations into account to improve large sample efficiency. For 3SLS, the 2SLS residuals are used to estimate the cross-equation error covariance matrix.

The SUR and 3SLS methods can be iterated by recomputing the estimate of the cross-equation covariance matrix from the SUR or 3SLS residuals and then computing new SUR or 3SLS estimates based on this updated covariance matrix estimate. Continuing this iteration until convergence produces ITSUR or IT3SLS estimates.

FIML Estimation Method

The FIML estimator is a system generalization of the LIML estimator. The FIML method involves minimizing the determinant of the covariance matrix associated with residuals of the reduced form of the equation system. From a maximum likelihood standpoint, the LIML method involves assuming that the errors are normally distributed and then maximizing the likelihood function subject to restrictions on a particular equation. FIML is similar, except that the likelihood function is maximized subject to restrictions on all of the parameters in the model, not just those in the equation being estimated.

Note: the RESTRICT, SRESTRICT, TEST, and STEST statements are not supported when the FIML method is used.

Choosing a Method for Simultaneous Equations

A number of factors should be taken into account in choosing an estimation method. Although system methods are asymptotically most efficient in the absence of specification error, system methods are more sensitive to specification error than single equation methods.

In practice, models are never perfectly specified. It is a matter of judgment whether the misspecification is serious enough to warrant avoidance of system methods.

Another factor to consider is sample size. With small samples, 2SLS may be preferred to 3SLS. In general, it is difficult to say much about the small sample properties of K-class estimators because this depends on the regressors used.

LIML and FIML are invariant to the normalization rule imposed but are computationally more expensive than 2SLS or 3SLS.

If the reason for contemporaneous correlation among errors across equations is a common omitted variable, it is not necessarily best to apply SUR. SUR parameter estimates are more sensitive to specification error than OLS. OLS may produce better parameter estimates under these circumstances. SUR estimates are also affected by the sampling variation of the error covariance matrix. There is some evidence from Monte Carlo studies that SUR is less efficient than OLS in small samples.

ANOVA Table for Instrumental Variables Methods

In the instrumental variables methods (2SLS, LIML, K-class, MELO), first-stage predicted values are substituted for the endogenous regressors. As a result, the regression sum of squares (RSS) and the error sum of squares (ESS) do not sum to the total corrected sum of squares for the dependent variable (TSS). The "Analysis of Variance" table printed for the second-stage results serves to display these sums of squares and the mean squares used for the *F*-test, but this table is not a variance decomposition in the usual analysis of variance sense.

The *F*-test shown in the instrumental variables case is a valid test of the no-regression hypothesis that the true coefficients of all regressors are 0. However, because of the first-stage projection of the regression mean square, this is a Wald-type test statistic, which is asymptotically *F* but not exactly F-distributed in finite samples. Thus, for small samples the *F*-test is only approximate when instrumental variables are used.

The R^2 Statistics

As explained in the section "ANOVA Table for Instrumental Variables Methods" on page 1505 when instrumental variables are used, the regression sum of squares (RSS) and the error sum of squares (ESS) do not sum to the total corrected sum of squares. In this case, there are several ways that the R^2 statistic can be defined.

The definition of R^2 used by the SYSLIN procedure is

$$\mathrm{R}^2 = \frac{\mathrm{RSS}}{\mathrm{RSS} + \mathrm{ESS}}$$

This definition is consistent with the *F*-test of the null hypothesis that the true coefficients of all regressors are zero. However, this R^2 may not be a good measure of the goodness of fit of the model.

System Weighted R^2 and System Weighted Mean Square Error

The system weighted R^2, printed for the 3SLS, IT3SLS, SUR, ITSUR, and FIML methods, is computed as follows.

$$\mathrm{R}^2 = \mathbf{Y}'\mathbf{W}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\mathbf{W}\mathbf{Y}/\mathbf{Y}'\mathbf{W}\mathbf{Y}$$

In this equation the matrix **X'X** is $\mathbf{R}'\mathbf{W}\mathbf{R}$, and **W** is the projection matrix of the instruments:

$$\mathbf{W} = \mathbf{S}^{-1} \otimes \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$$

The matrix **Z** is the instrument set, **R** is the the regressor set, and **S** is the estimated cross-model covariance matrix.

The system weighted MSE, printed for the 3SLS, IT3SLS, SUR, ITSUR, and FIML methods, is computed as follows:

$$MSE = \frac{1}{tdf}(\mathbf{Y'WY} - \mathbf{Y'WR}(\mathbf{X'X})^{-1}\mathbf{R'WY})$$

In this equation *tdf* is the sum of the error degrees of freedom for the equations in the system.

Computational Details

This section discusses various computational details.

Computation of Least Squares-Based Estimators

Let the system be composed of G equations, and the ith equation be expressed in this form:

$$\mathbf{y}_i = Y_i \boldsymbol{\beta}_i + X_i \gamma_i + \mathbf{u}$$

where

$\mathbf{y}_i$	is the vector of observations on the dependent variable
Y_i	is the matrix of observations on the endogenous variables included in the equation
$\boldsymbol{\beta}_i$	is the vector of parameters associated with Y_i
X_i	is the matrix of observations on the predetermined variables included in the equation
γ_i	is the vector of parameters associated with X_i
$\mathbf{u}$	is a vector of errors

Let $\hat{V}_i = Y_i - \hat{Y}_i$, where $\hat{Y}_i$ is the projection of Y_i onto the space spanned by the instruments matrix Z.

Let

$$\boldsymbol{\delta}_i = \left[\begin{array}{c} \boldsymbol{\beta}_i \\ \gamma_i \end{array} \right]$$

be the vector of parameters associated with both the endogenous and exogenous variables.

The K class of estimators (Theil 1971) is defined by

$$\hat{\boldsymbol{\delta}}_{i,k} = \left[\begin{array}{cc} Y_i'Y_i - k\hat{V}_i'\hat{V}_i & Y_i'X_i \\ X_i'Y_i & X_i'X_i \end{array} \right]^{-1} \left[\begin{array}{c} (Y_i - kV_i)'y_i \\ X_i'y_i \end{array} \right]$$

where k is a user-defined value.

Let

$$R = [Y_i \quad X_i]$$

and

$$\hat{R} = [\hat{Y}_i \quad X_i]$$

The 2SLS estimator is defined as

$$\hat{\boldsymbol{\delta}}_{i,2SLS} = [\hat{R}_i' \, \hat{R}_i]^{-1} \hat{R}_i' y_i$$

Let $\mathbf{y}$ and $\boldsymbol{\delta}$ be the vectors obtained by stacking the vectors of dependent variables and parameters for all G equations, and let R and $\hat{R}$ be the block diagonal matrices formed by R_i and $\hat{R}_i$, respectively.

The SUR and ITSUR estimators are defined as

$$\hat{\boldsymbol{\delta}}_{(IT)SUR} = \left[R' \left(\hat{\Sigma}^{-1} \otimes I\right) R\right]^{-1} R' \left(\hat{\Sigma}^{-1} \otimes I\right) \mathbf{y}$$

while the 3SLS and IT3SLS estimators are defined as

$$\hat{\boldsymbol{\delta}}_{(IT)3SLS} = \left[\hat{R}' \left(\hat{\Sigma}^{-1} \otimes I\right) \hat{R}\right]^{-1} \hat{R}' \left(\hat{\Sigma}^{-1} \otimes I\right) \mathbf{y}$$

where I is the identity matrix, and $\hat{\Sigma}$ is an estimator of the cross-equation correlation matrix. For 3SLS, $\hat{\Sigma}$ is obtained from the 2SLS estimation, while for SUR it is derived from the OLS estimation. For IT3SLS and ITSUR, it is obtained iteratively from the previous estimation step, until convergence.

Computation of Standard Errors

The VARDEF= option in the PROC SYSLIN statement controls the denominator used in calculating the cross-equation covariance estimates and the parameter standard errors and covariances. The values of the VARDEF= option and the resulting denominator are as follows:

N	uses the number of nonmissing observations.
DF	uses the number of nonmissing observations less the degrees of freedom in the model.
WEIGHT	uses the sum of the observation weights given by the WEIGHTS statement.
WDF	uses the sum of the observation weights given by the WEIGHTS statement less the degrees of freedom in the model.

The VARDEF= option does not affect the model mean square error, root mean square error, or R^2 statistics. These statistics are always based on the error degrees of freedom, regardless of the VARDEF= option. The VARDEF= option also does not affect the dependent variable coefficient of variation (C.V.).

Reduced-Form Estimates

The REDUCED option on the PROC SYSLIN statement computes estimates of the reduced-form coefficients. The REDUCED option requires that the equation system be square. If there are fewer models than endogenous variables, IDENTITY statements can be used to complete the equation system.

The reduced-form coefficients are computed as follows. Represent the equation system, with all endogenous variables moved to the left-hand side of the equations and identities, as

$$\mathbf{BY} = \mathbf{\Gamma X}$$

Here **B** is the estimated coefficient matrix for the endogenous variables **Y**, and **Γ** is the estimated coefficient matrix for the exogenous (or predetermined) variables **X**.

The system can be solved for **Y** as follows, provided **B** is square and nonsingular:

$$\mathbf{Y} = \mathbf{B}^{-1}\mathbf{\Gamma X}$$

The reduced-form coefficients are the matrix $\mathbf{B}^{-1}\mathbf{\Gamma}$.

Uncorrelated Errors Across Equations

The SDIAG option in the PROC SYSLIN statement computes estimates assuming uncorrelated errors across equations. As a result, when the SDIAG option is used, the 3SLS estimates are identical to 2SLS estimates, and the SUR estimates are the same as the OLS estimates.

Over Identification Restrictions

The OVERID option in the MODEL statement can be used to test for over identifying restrictions on parameters of each equation. The null hypothesis is that the predetermined variables not appearing in any equation have zero coefficients. The alternative hypothesis is that at least one of the assumed zero coefficients is nonzero. The test is approximate and rejects the null hypothesis too frequently for small sample sizes.

The formula for the test is given as follows. Let $y_i = \beta_i \mathbf{Y}_i + \gamma_i \mathbf{Z}_i + e_i$ be the *i*th equation. $\mathbf{Y}_i$ are the endogenous variables that appear as regressors in the *i*th equation, and $\mathbf{Z}_i$ are the instrumental variables that appear as regressors in the *i*th equation. Let N_i be the number of variables in $\mathbf{Y}_i$ and $\mathbf{Z}_i$.

Let $v_i = y_i - \mathbf{Y}_i\hat{\beta}_i$. Let **Z** represent all instrumental variables, *T* be the total number of observations, and *K* be the total number of instrumental variables. Define $\hat{l}$ as follows:

$$\hat{l} = \frac{v'_i(\mathbf{I} - \mathbf{Z}_i(\mathbf{Z}'_i\mathbf{Z}_i)^{-1}\mathbf{Z}'_i)v_i}{v'_i(\mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}')v_i}$$

Then the test statistic

$$\frac{T-K}{K-N_i}(\hat{l}-1)$$

is distributed approximately as an F with $K-N_i$ and $T-K$ degrees of freedom. Refer to Basmann (1960) for more information.

Fuller's Modification to LIML

The ALPHA= option in the PROC SYSLIN and MODEL statements parameterizes Fuller's modification to LIML. This modification is $k = \gamma - (\alpha/(n-g))$, where α is the value of the ALPHA= option, γ is the LIML *k* value, *n* is the number of observations, and *g* is the number of predetermined variables. Fuller's modification is not used unless the ALPHA= option is specified. Refer to Fuller (1977) for more information.

Missing Values

Observations having a missing value for any variable in the analysis are excluded from the computations.

OUT= Data Set

The output SAS data set produced by the OUT= option in the PROC SYSLIN statement contains all the variables in the input data set and the variables containing predicted values and residuals specified by OUTPUT statements.

The residuals are computed as actual values minus predicted values. Predicted values never use lags of other predicted values, as would be desirable for dynamic simulation. For these applications, PROC SIMLIN is available to predict or simulate values from the estimated equations.

OUTEST= Data Set

The OUTEST= option produces a TYPE=EST output SAS data set containing estimates from the regressions. The variables in the OUTEST= data set are as follows:

BY variables	the BY statement variables are included in the OUTEST= data set
TYPE	identifies the estimation type for the observations. The _TYPE_ value INST indicates first-stage regression estimates. Other values indicate the estimation method used: 2SLS indicates two-stage least squares results, 3SLS indicates three-stage least squares results, LIML indicates limited information maximum likelihood results, and so forth. Observations added by IDENTITY statements have the _TYPE_ value IDENTITY.
MODEL	the model label. The model label is the label specified on the MODEL statement or the dependent variable name if no label is specified. For first-stage regression estimates, _MODEL_ has the value FIRST.

DEPVAR	the name of the dependent variable for the model
NAME	the names of the regressors for the rows of the covariance matrix, if the COVOUT option is specified. _NAME_ has a blank value for the parameter estimates observations. The _NAME_ variable is not included in the OUTEST= data set unless the COVOUT option is used to output the covariance of parameter estimates matrix.
SIGMA	contains the root mean square error for the model, which is an estimate of the standard deviation of the error term. The _SIGMA_ variable contains the same values reported as Root MSE in the printed output.
INTERCEPT	the intercept parameter estimates
regressors	the regressor variables from all the MODEL statements are included in the OUTEST= data set. Variables used in IDENTIFY statements are also included in the OUTEST= data set.

The parameter estimates are stored under the names of the regressor variables. The intercept parameters are stored in the variable INTERCEP. The dependent variable of the model is given a coefficient of -1. Variables not in a model have missing values for the OUTEST= observations for that model.

Some estimation methods require computation of preliminary estimates. All estimates computed are output to the OUTEST= data set. For each BY group and each estimation, the OUTEST= data set contains one observation for each MODEL or IDENTITY statement. Results for different estimations are identified by the _TYPE_ variable.

For example, consider the following statements:

```
proc syslin data=a outest=est 3sls;
   by b;
   endogenous y1 y2;
   instruments x1-x4;
   model y1 = y2 x1 x2;
   model y2 = y1 x3 x4;
   identity x1 = x3 + x4;
run;
```

The 3SLS method requires both a preliminary 2SLS stage and preliminary first stage regressions for the endogenous variable. The OUTEST= data set thus contains 3 different kinds of estimates. The observations for the first-stage regression estimates have the _TYPE_ value INST. The observations for the 2SLS estimates have the _TYPE_ value 2SLS. The observations for the final 3SLS estimates have the _TYPE_ value 3SLS.

Since there are 2 endogenous variables in this example, there are 2 first-stage regressions and 2 _TYPE_=INST observations in the OUTEST= data set. Since there are 2 model statements, there are 2 OUTEST= observations with _TYPE_=2SLS and 2 observations with _TYPE_=3SLS. In addition, the OUTEST= data set contains an observation with the _TYPE_ value IDENTITY containing the coefficients specified by the IDENTITY statement. All these observations are repeated for each BY-group in the input data set defined by the values of the BY variable B.

When the COVOUT option is specified, the estimated covariance matrix for the parameter estimates is included in the OUTEST= data set. Each observation for parameter estimates is followed by observations containing the rows of the parameter covariance matrix for that model. The row of the covariance matrix is identified by the variable _NAME_. For observations that contain parameter estimates, _NAME_ is blank. For covariance observations, _NAME_ contains the regressor name for the row of the covariance matrix, and the regressor variables contain the covariances.

See Example 26.1 for an example of the OUTEST= data set.

OUTSSCP= Data Set

The OUTSSCP= option produces a TYPE=SSCP output SAS data set containing sums of squares and cross products. The data set contains all variables used in the MODEL, IDENTITY, and VAR statements. Observations are identified by the variable _NAME_.

The OUTSSCP= data set can be useful when a large number of observations are to be explored in many different SYSLIN runs. The sum-of-squares-and-crossproducts matrix can be saved with the OUTSSCP= option and used as the DATA= data set on subsequent SYSLIN runs. This is much less expensive computationally because PROC SYSLIN never reads the original data again. In the step that creates the OUTSSCP= data set, include in the VAR statement all the variables you expect to use.

Printed Output

The printed output produced by the SYSLIN procedure is as follows:

1. If the SIMPLE option is used, a table of descriptive statistics is printed showing the sum, mean, sum of squares, variance, and standard deviation for all the variables used in the models.
2. First-stage regression results are printed if the FIRST option is specified and an instrumental variables method is used. This shows the regression of each endogenous variable on the variables in the INSTRUMENTS list.
3. The results of the second-stage regression are printed for each model. (See “Printed Output for Each Model,” which follows.)
4. If a systems method like 3SLS, SUR, or FIML is used, the cross-equation error covariance matrix is printed. This matrix is shown four ways: the covariance matrix itself, the correlation matrix form, the inverse of the correlation matrix, and the inverse of the covariance matrix.
5. If a systems method like 3SLS, SUR, or FIML is used, the system weighted mean square error and system weighted R^2 statistics are printed. The system weighted MSE and R^2 measure the fit of the joint model obtained by stacking

all the models together and performing a single regression with the stacked observations weighted by the inverse of the model error variances.

6. If a systems method like 3SLS, SUR, or FIML is used, the final results are printed for each model.
7. If the REDUCED option is used, the reduced-form coefficients are printed. This consists of the structural coefficient matrix for the endogenous variables, the structural coefficient matrix for the exogenous variables, the inverse of the endogenous coefficient matrix, and the reduced-form coefficient matrix. The reduced-form coefficient matrix is the product of the inverse of the endogenous coefficient matrix and the exogenous structural coefficient matrix.

Printed Output for Each Model

The results printed for each model include the “Analysis of Variance” table, the “Parameter Estimates” table, and optional items requested by TEST statements or by options on the MODEL statement.

The printed output produced for each model is described in the following.

The Analysis of Variance table includes the following:

- the model degrees of freedom, sum of squares, and mean square
- the error degrees of freedom, sum of squares, and mean square. The error mean square is computed by dividing the error sum of squares by the error degrees of freedom and is not effected by the VARDEF= option.
- the corrected total degrees of freedom and total sum of squares. Note that for instrumental variables methods the model and error sums of squares do not add to the total sum of squares.
- the *F*-ratio, labeled “F Value,” and its significance, labeled “PROB>F,” for the test of the hypothesis that all the nonintercept parameters are 0
- the root mean square error. This is the square root of the error mean square.
- the dependent variable mean
- the coefficient of variation (C.V.) of the dependent variable
- the R^2 statistic. This R^2 is computed consistently with the calculation of the *F* statistic. It is valid for hypothesis tests but may not be a good measure of fit for models estimated by instrumental variables methods.
- the R^2 statistic adjusted for model degrees of freedom, labeled “Adj R-SQ”

The Parameter Estimates table includes the following.

- estimates of parameters for regressors in the model and the Lagrangian parameter for each restriction specified
- a degrees of freedom column labeled DF. Estimated model parameters have 1 degree of freedom. Restrictions have a DF of -1. Regressors or restrictions dropped from the model due to collinearity have a DF of 0.

- the standard errors of the parameter estimates
- the t statistics, which are the parameter estimates divided by the standard errors
- the significance of the *t*-tests for the hypothesis that the true parameter is 0, labeled "Pr > |t|." As previously noted, the significance tests are strictly valid in finite samples only for OLS estimates but are asymptotically valid for the other methods.
- the standardized regression coefficients, if the STB option is specified. This is the parameter estimate multiplied by the ratio of the standard deviation of the regressor to the standard deviation of the dependent variable.
- the labels of the regressor variables or restriction labels

In addition to the Analysis of Variance table and the Parameter Estimates table, the results printed for each model may include the following:

1. If TEST statements are specified, the test results are printed.
2. If the DW option is specified, the Durbin-Watson statistic and first-order autocorrelation coefficient are printed.
3. If the OVERID option is specified, the results of Basmann's test for overidentifying restrictions are printed.
4. If the PLOT option is used, plots of residual against each regressor are printed.
5. If the COVB or CORRB options are specified, the results for each model also include the covariance or correlation matrix of the parameter estimates. For systems methods like 3SLS and FIML, the COVB and CORB output is printed for the whole system after the output for the last model, instead of separately for each model.

The third stage output for 3SLS, SUR, IT3SLS, ITSUR, and FIML does not include the Analysis of Variance table. When a systems method is used, the second stage output does not include the optional output, except for the COVB and CORB matrices.

ODS Table Names

PROC SYSLIN assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 26.1. ODS Tables Produced in PROC SYSLIN

ODS Table Name	Description	Option
ANOVA	Summary of the SSE, MSE for the equations	default
AugXPXMat	Model Crossproducts	XPX
AutoCorrStat	Autocorrelation Statistics	default
ConvCrit	Convergence criteria for estimation	default
ConvergenceStatus	Convergence status	default
CorrB	Correlations of parameters	CORRB
CorrResiduals	Correlations of residuals	CORRS
CovB	Covariance of parameters	COVB
CovResiduals	Covariance of residuals	
EndoMat	Endogenous Variables	
Equations	Listing of equations to estimate	default
ExogMat	Exogenous Variables	
FitStatistics	Statistics of Fit	default
InvCorrResiduals	Inverse Correlations of residuals	CORRS
InvCovResiduals	InvCovariance of residuals	COVS
InvEndoMat	Inverse Endogenous Variables	
InvXPX	$X'X$ inverse for System	I
IterHistory	Iteration printing	ITALL/ITPRINT
MissingValues	Missing values generated by the program	default
ModelVars	Name and label for the Model	default
ParameterEstimates	Parameter Estimates	default
RedMat	Reduced Form	REDUCED
SimpleStatistics	Descriptive statistics	SIMPLE
SSCP	Model Crossproducts	
TestResults	Test for Overidentifying Restrictions	
Weight	Weighted Model Statistics	
YPY	Y'Y matrices	USSCP2

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the SYSLIN procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

ODS Graph Names

PROC SYSLIN assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 26.2.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 26.2. ODS Graphics Produced by PROC SYSLIN

ODS Graph Name	Plot Description
ActualByPredicted	Predicted vs actual plot
QQPlot	QQ plot of residuals
ResidualHistogram	Histogram of the residuals
ResidualPlot	Residual plot

Examples

Example 26.1. Klein's Model I Estimated with LIML and 3SLS

This example uses PROC SYSLIN to estimate the classic Klein Model I. For a discussion of this model, see Theil (1971). The following statements read the data.

```
*---------------------------Klein's Model I---------------------------*
| By L.R. Klein, Economic Fluctuations in the United States, 1921-1941 |
| (1950), NY: John Wiley.   A macro-economic model of the U.S. with    |
| three behavioral equations, and several identities. See Theil, p.456.|
*----------------------------------------------------------------------*;
data klein;
input year c p w i x wp g t k wsum;
   date=mdy(1,1,year);
   format date monyy.;
   y   =c+i+g-t;
   yr  =year-1931;
   klag=lag(k);
   plag=lag(p);
   xlag=lag(x);
   label year='Year'
         date='Date'
         c   ='Consumption'
         p   ='Profits'
         w   ='Private Wage Bill'
         i   ='Investment'
         k   ='Capital Stock'
         y   ='National Income'
         x   ='Private Production'
         wsum='Total Wage Bill'
         wp  ='Govt Wage Bill'
         g   ='Govt Demand'
         i   ='Taxes'
         klag='Capital Stock Lagged'
         plag='Profits Lagged'
         xlag='Private Product Lagged'
         yr  ='YEAR-1931';
   datalines;
1920     .  12.7     .    .  44.9    .     .     .  182.8     .
1921  41.9  12.4  25.5 -0.2  45.6  2.7   3.9   7.7  182.6  28.2
1922  45.0  16.9  29.3  1.9  50.1  2.9   3.2   3.9  184.5  32.2
1923  49.2  18.4  34.1  5.2  57.2  2.9   2.8   4.7  189.7  37.0
1924  50.6  19.4  33.9  3.0  57.1  3.1   3.5   3.8  192.7  37.0
1925  52.6  20.1  35.4  5.1  61.0  3.2   3.3   5.5  197.8  38.6
1926  55.1  19.6  37.4  5.6  64.0  3.3   3.3   7.0  203.4  40.7
1927  56.2  19.8  37.9  4.2  64.4  3.6   4.0   6.7  207.6  41.5
1928  57.3  21.1  39.2  3.0  64.5  3.7   4.2   4.2  210.6  42.9
```

```
   1929  57.8  21.7  41.3  5.1  67.0  4.0   4.1   4.0  215.7  45.3
   1930  55.0  15.6  37.9  1.0  61.2  4.2   5.2   7.7  216.7  42.1
   1931  50.9  11.4  34.5 -3.4  53.4  4.8   5.9   7.5  213.3  39.3
   1932  45.6   7.0  29.0 -6.2  44.3  5.3   4.9   8.3  207.1  34.3
   1933  46.5  11.2  28.5 -5.1  45.1  5.6   3.7   5.4  202.0  34.1
   1934  48.7  12.3  30.6 -3.0  49.7  6.0   4.0   6.8  199.0  36.6
   1935  51.3  14.0  33.2 -1.3  54.4  6.1   4.4   7.2  197.7  39.3
   1936  57.7  17.6  36.8  2.1  62.7  7.4   2.9   8.3  199.8  44.2
   1937  58.7  17.3  41.0  2.0  65.0  6.7   4.3   6.7  201.8  47.7
   1938  57.5  15.3  38.2 -1.9  60.9  7.7   5.3   7.4  199.9  45.9
   1939  61.6  19.0  41.6  1.3  69.5  7.8   6.6   8.9  201.2  49.4
   1940  65.0  21.1  45.0  3.3  75.7  8.0   7.4   9.6  204.5  53.0
   1941  69.7  23.5  53.3  4.9  88.4  8.5  13.8  11.6  209.4  61.8
   ;
   run;
```

The following statements estimate the Klein model using the limited information maximum likelihood method. In addition, the parameter estimates are written to a SAS data set with the OUTEST= option.

```
proc syslin data=klein outest=b liml;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model c = p plag  wsum;
   invest:  model i = p plag  klag;
   labor:   model w = x xlag  yr;
run;


proc print data=b; run;
```

The PROC SYSLIN estimates are shown in Output 26.1.1 through Output 26.1.3.

Output 26.1.1. LIML Estimates for Consumption

```
                              The SYSLIN Procedure
               Limited-Information Maximum Likelihood Estimation

                       Model                       CONSUME
                       Dependent Variable                c
                       Label                   Consumption


                              Analysis of Variance

                                    Sum of         Mean
     Source                 DF     Squares       Square    F Value    Pr > F

     Model                   3    854.3541     284.7847     118.42    <.0001
     Error                  17    40.88419     2.404952
     Corrected Total        20    941.4295


             Root MSE            1.55079    R-Square       0.95433
             Dependent Mean     53.99524    Adj R-Sq       0.94627
             Coeff Var           2.87209


                              Parameter Estimates

                   Parameter  Standard                         Variable
Variable       DF   Estimate     Error  t Value  Pr > |t|  Label

Intercept       1   17.14765  2.045374     8.38    <.0001  Intercept
p               1   -0.22251  0.224230    -0.99    0.3349  Profits
plag            1   0.396027  0.192943     2.05    0.0558  Profits Lagged
wsum            1   0.822559  0.061549    13.36    <.0001  Total Wage Bill
```

Output 26.1.2. LIML Estimates for Investments

```
                Limited-Information Maximum Likelihood Estimation

                          Model                    INVEST
                          Dependent Variable            i
                          Label                     Taxes


                               Analysis of Variance

                                     Sum of        Mean
      Source                 DF     Squares      Square    F Value    Pr > F

      Model                   3    210.3790    70.12634      34.06    <.0001
      Error                  17    34.99649    2.058617
      Corrected Total        20    252.3267


              Root MSE             1.43479    R-Square       0.85738
              Dependent Mean       1.26667    Adj R-Sq       0.83221
              Coeff Var          113.27274


                               Parameter Estimates

                     Parameter  Standard                       Variable
Variable       DF     Estimate     Error  t Value  Pr > |t|  Label

Intercept       1     22.59083  9.498146     2.38    0.0294  Intercept
p               1     0.075185  0.224712     0.33    0.7420  Profits
plag            1     0.680386  0.209145     3.25    0.0047  Profits Lagged
klag            1     -0.16826  0.045345    -3.71    0.0017  Capital Stock Lagged
```

Output 26.1.3. LIML Estimates for Labor

```
               Limited-Information Maximum Likelihood Estimation

                    Model                             LABOR
                    Dependent Variable                    w
                    Label                 Private Wage Bill


                              Analysis of Variance

                                    Sum of        Mean
     Source                DF      Squares      Square    F Value    Pr > F

     Model                  3     696.1485    232.0495     393.62    <.0001
     Error                 17     10.02192    0.589525
     Corrected Total       20     794.9095


             Root MSE             0.76781    R-Square       0.98581
             Dependent Mean      36.36190    Adj R-Sq       0.98330
             Coeff Var            2.11156


                              Parameter Estimates

                   Parameter  Standard                        Variable
Variable       DF   Estimate     Error  t Value  Pr > |t|  Label

Intercept       1   1.526187  1.320838     1.16    0.2639  Intercept
x               1   0.433941  0.075507     5.75    <.0001  Private Production
xlag            1   0.151321  0.074527     2.03    0.0583  Private Product
                                                           Lagged
yr              1   0.131593  0.035995     3.66    0.0020  YEAR-1931
```

The OUTEST= data set is shown in part in Output 26.1.4. Note that the data set contains the parameter estimates and root mean square errors, _SIGMA_, for the first stage instrumental regressions as well as the parameter estimates and σ for the LIML estimates for the three structural equations.

Output 26.1.4. The OUTEST= Data Set

```
  Obs _TYPE_  _STATUS_   _MODEL_ _DEPVAR_ _SIGMA_ Intercept   klag     plag

   1   LIML  0 Converged CONSUME    c     1.55079  17.1477     .      0.39603
   2   LIML  0 Converged INVEST     i     1.43479  22.5908  -0.16826 0.68039
   3   LIML  0 Converged LABOR      w     0.76781   1.5262     .        .

  Obs   xlag   wp  g  t     yr     c      p      w   i     x       wsum   k  y

   1   .        .  .  .     .      -1  -0.22251   .   .   .       0.82256  .  .
   2   .        .  .  .     .       .   0.07518   .  -1   .          .     .  .
   3  0.15132   .  .  .  0.13159   .     .       -1   .  0.43394    .     .  .
```

The following statements estimate the model using the 3SLS method. The reduced-form estimates are produced by the REDUCED option; IDENTITY statements are used to make the model complete.

```
proc syslin data=klein 3sls reduced;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model    c = p plag wsum;
   invest:  model    i = p plag klag;
   labor:   model    w = x xlag yr;
   product: identity x = c + i + g;
   income:  identity y = c + i + g - t;
   profit:  identity p = y - w;
   stock:   identity k = klag + i;
   wage:    identity wsum = w + wp;
run;
```

The preliminary 2SLS results and estimated cross-model covariance matrix are not shown. The 3SLS estimates are shown in Output 26.1.5 through Output 26.1.7. The reduced-form estimates are shown in Output 26.1.8 through Output 26.1.11.

Output 26.1.5. 3SLS Estimates for Consumption

```
                                The SYSLIN Procedure
                         Three-Stage Least Squares Estimation

                        System Weighted MSE              5.9342
                        Degrees of freedom                   51
                        System Weighted R-Square         0.9550


                          Model                        CONSUME
                          Dependent Variable                 c
                          Label                    Consumption


                                Parameter Estimates

                     Parameter  Standard                      Variable
Variable       DF     Estimate     Error  t Value  Pr > |t|  Label

Intercept       1     16.44079  1.449925    11.34    <.0001  Intercept
p               1     0.124890  0.120179     1.04    0.3133  Profits
plag            1     0.163144  0.111631     1.46    0.1621  Profits Lagged
wsum            1     0.790081  0.042166    18.74    <.0001  Total Wage Bill
```

Output 26.1.6. 3SLS Estimates for Investments

```
                       Three-Stage Least Squares Estimation

                          Model                   INVEST
                          Dependent Variable           i
                          Label                    Taxes


                               Parameter Estimates

                    Parameter  Standard                        Variable
Variable       DF    Estimate     Error  t Value  Pr > |t|  Label

Intercept       1    28.17785  7.550853     3.73    0.0017  Intercept
p               1    -0.01308  0.179938    -0.07    0.9429  Profits
plag            1    0.755724  0.169976     4.45    0.0004  Profits Lagged
klag            1    -0.19485  0.036156    -5.39    <.0001  Capital Stock Lagged
```

Output 26.1.7. 3SLS Estimates for Labor

```
                       Three-Stage Least Squares Estimation

                     Model                             LABOR
                     Dependent Variable                    w
                     Label                     Private Wage Bill
```

Output 26.1.8. Reduced-Form Estimates

```
                      Three-Stage Least Squares Estimation

                              Endogenous Variables

                           c             p             w             i

         CONSUME           1      -0.12489             0             0
         INVEST            0      0.013079             0             1
         LABOR             0             0             1             0
         PRODUCT          -1             0             0            -1
         INCOME           -1             0             0            -1
         PROFIT            0             1             1             0
         STOCK             0             0             0            -1
         WAGE              0             0            -1             0

                              Endogenous Variables

                           x          wsum             k             y

         CONSUME           0      -0.79008             0             0
         INVEST            0             0             0             0
         LABOR      -0.40049             0             0             0
         PRODUCT           1             0             0             0
         INCOME            0             0             0             1
         PROFIT            0             0             0            -1
         STOCK             0             0             1             0
         WAGE              0             1             0             0
```

Output 26.1.9. Reduced-Form Estimates

```
                      Three-Stage Least Squares Estimation

                              Exogenous Variables

                   Intercept          plag          klag          xlag

        CONSUME     16.44079      0.163144             0             0
        INVEST      28.17785      0.755724      -0.19485             0
        LABOR       1.797218             0             0      0.181291
        PRODUCT            0             0             0             0
        INCOME             0             0             0             0
        PROFIT             0             0             0             0
        STOCK              0             0             1             0
        WAGE               0             0             0             0

                              Exogenous Variables

                          yr             g             t            wp

         CONSUME           0             0             0             0
         INVEST            0             0             0             0
         LABOR      0.149674             0             0             0
         PRODUCT           0             1             0             0
         INCOME            0             1            -1             0
         PROFIT            0             0             0             0
         STOCK             0             0             0             0
         WAGE              0             0             0             1
```

Output 26.1.10. Reduced-Form Estimates

```
                     Three-Stage Least Squares Estimation

                         Inverse Endogenous Variables

                    CONSUME        INVEST         LABOR       PRODUCT

           c       1.634654      0.634654      1.095657      0.438802
           p       0.972364      0.972364      -0.34048      -0.13636
           w       0.649572      0.649572      1.440585      0.576943
           i       -0.01272      0.987282      0.004453      0.001783
           x       1.621936      1.621936       1.10011      1.440585
           wsum    0.649572      0.649572      1.440585      0.576943
           k       -0.01272      0.987282      0.004453      0.001783
           y       1.621936      1.621936       1.10011      0.440585

                         Inverse Endogenous Variables

                     INCOME        PROFIT         STOCK          WAGE

           c       0.195852      0.195852       9.2E-17      1.291509
           p       1.108721      1.108721      5.51E-17      0.768246
           w       0.072629      0.072629      3.68E-17      0.513215
           i        -0.0145       -0.0145      1.85E-20      -0.01005
           x       0.181351      0.181351       9.2E-17      1.281461
           wsum    0.072629      0.072629      1.08E-16      1.513215
           k        -0.0145       -0.0145             1      -0.01005
           y       1.181351      0.181351       9.2E-17      1.281461
```

Output 26.1.11. Reduced-Form Estimates

```
                     Three-Stage Least Squares Estimation

                                 Reduced Form

                  Intercept          plag          klag          xlag

          c         46.7273      0.746307      -0.12366      0.198633
          p        42.77363      0.893474      -0.18946      -0.06173
          w        31.57207      0.596871      -0.12657      0.261165
          i         27.6184      0.744038      -0.19237      0.000807
          x         74.3457      1.490345      -0.31603       0.19944
          wsum     31.57207      0.596871      -0.12657      0.261165
          k         27.6184      0.744038       0.80763      0.000807
          y         74.3457      1.490345      -0.31603       0.19944

                                  Reduced Form

                         yr             g             t            wp

           c       0.163991      0.634654      -0.19585      1.291509
           p       -0.05096      0.972364      -1.10872      0.768246
           w       0.215618      0.649572      -0.07263      0.513215
           i       0.000667      -0.01272      0.014501      -0.01005
           x       0.164658      1.621936      -0.18135      1.281461
           wsum    0.215618      0.649572      -0.07263      1.513215
           k       0.000667      -0.01272      0.014501      -0.01005
           y       0.164658      1.621936      -1.18135      1.281461
```

Example 26.2. Grunfeld’s Model Estimated with SUR

The following example was used by Zellner in his classic 1962 paper on seemingly unrelated regressions. Different stock prices often move in the same direction at a given point in time. The SUR technique may provide more efficient estimates than OLS in this situation.

The following statements read the data. (The prefix GE stands for General Electric and WH stands for Westinghouse.)

```
*---------Zellner’s Seemingly Unrelated Technique-----------*
| A. Zellner, "An Efficient Method of Estimating Seemingly   |
| Unrelated Regressions and Tests for Aggregation Bias,"     |
| JASA 57(1962) pp.348-364                                   |
|                                                            |
| J.C.G. Boot, "Investment Demand: an Empirical Contribution |
| to the Aggregation Problem," IER 1(1960) pp.3-30.          |
|                                                            |
| Y. Grunfeld, "The Determinants of Corporate Investment,"   |
| Unpublished thesis, Chicago, 1958                          |
*------------------------------------------------------------*;

data grunfeld;
   input year ge_i ge_f ge_c wh_i wh_f wh_c;
   label ge_i = ’Gross Investment, GE’
         ge_c = ’Capital Stock Lagged, GE’
         ge_f = ’Value of Outstanding Shares Lagged, GE’
         wh_i = ’Gross Investment, WH’
         wh_c = ’Capital Stock Lagged, WH’
         wh_f = ’Value of Outstanding Shares Lagged, WH’;
   datalines;
1935     33.1      1170.6    97.8      12.93     191.5     1.8
1936     45.0      2015.8    104.4     25.90     516.0     .8
1937     77.2      2803.3    118.0     35.05     729.0     7.4
1938     44.6      2039.7    156.2     22.89     560.4     18.1
1939     48.1      2256.2    172.6     18.84     519.9     23.5
1940     74.4      2132.2    186.6     28.57     628.5     26.5
1941     113.0     1834.1    220.9     48.51     537.1     36.2
1942     91.9      1588.0    287.8     43.34     561.2     60.8
1943     61.3      1749.4    319.9     37.02     617.2     84.4
1944     56.8      1687.2    321.3     37.81     626.7     91.2
1945     93.6      2007.7    319.6     39.27     737.2     92.4
1946     159.9     2208.3    346.0     53.46     760.5     86.0
1947     147.2     1656.7    456.4     55.56     581.4     111.1
1948     146.3     1604.4    543.4     49.56     662.3     130.6
1949     98.3      1431.8    618.3     32.04     583.8     141.8
1950     93.5      1610.5    647.4     32.24     635.2     136.7
1951     135.2     1819.4    671.3     54.38     723.8     129.7
1952     157.3     2079.7    726.1     71.78     864.1     145.5
1953     179.5     2371.6    800.3     90.08     1193.5    174.8
1954     189.6     2759.9    888.9     68.60     1188.9    213.5
;
```

The following statements compute the SUR estimates for the Grunfeld model.

```
proc syslin data=grunfeld sur;
   ge:      model ge_i = ge_f ge_c;
```

```
      westing: model wh_i = wh_f wh_c;
   run;
```

The PROC SYSLIN output is shown in Output 26.2.1.

Output 26.2.1. PROC SYSLIN Output for SUR

```
                              The SYSLIN Procedure
                         Ordinary Least Squares Estimation

                     Model                                   GE
                     Dependent Variable                    ge_i
                     Label                   Gross Investment, GE


                              Analysis of Variance

                                       Sum of        Mean
     Source                   DF      Squares      Square    F Value    Pr > F

     Model                     2     31632.03    15816.02      20.34    <.0001
     Error                    17     13216.59    777.4463
     Corrected Total          19     44848.62


             Root MSE            27.88272    R-Square       0.70531
             Dependent Mean     102.29000    Adj R-Sq       0.67064
             Coeff Var           27.25850


                              Parameter Estimates

                   Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -9.95631 31.37425   -0.32   0.7548 Intercept
ge_f           1  0.026551 0.015566    1.71   0.1063 Value of Outstanding Shares
                                                     Lagged, GE
ge_c           1  0.151694 0.025704    5.90   <.0001 Capital Stock Lagged, GE
```

```
                            The SYSLIN Procedure
                       Ordinary Least Squares Estimation

                   Model                              WESTING
                   Dependent Variable                    wh_i
                   Label                  Gross Investment, WH


                            Analysis of Variance

                                      Sum of        Mean
     Source                 DF       Squares      Square    F Value    Pr > F

     Model                   2      5165.553    2582.776      24.76    <.0001
     Error                  17      1773.234    104.3079
     Corrected Total        19      6938.787


             Root MSE            10.21312    R-Square       0.74445
             Dependent Mean      42.89150    Adj R-Sq       0.71438
             Coeff Var           23.81153


                            Parameter Estimates

                 Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -0.50939 8.015289   -0.06   0.9501 Intercept
wh_f           1  0.052894 0.015707    3.37   0.0037 Value of Outstanding Shares
                                                     Lagged, WH
wh_c           1  0.092406 0.056099    1.65   0.1179 Capital Stock Lagged, WH
```

```
                      The SYSLIN Procedure
           Seemingly Unrelated Regression Estimation

                     Cross Model Covariance

                            GE            WESTING

             GE            777.446       207.587
             WESTING       207.587       104.308


                     Cross Model Correlation

                            GE            WESTING

             GE            1.00000       0.72896
             WESTING       0.72896       1.00000


                 Cross Model Inverse Correlation

                            GE            WESTING

             GE            2.13397      -1.55559
             WESTING      -1.55559       2.13397


                  Cross Model Inverse Covariance

                            GE            WESTING

             GE           0.002745      -.005463
             WESTING      -.005463      0.020458
```

```
                      The SYSLIN Procedure
           Seemingly Unrelated Regression Estimation

             System Weighted MSE            0.9719
             Degrees of freedom                 34
             System Weighted R-Square       0.6284


           Model                                   GE
           Dependent Variable                    ge_i
           Label                   Gross Investment, GE


                      Parameter Estimates

                 Parameter Standard                    Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -27.7193 29.32122   -0.95   0.3577 Intercept
ge_f           1  0.038310 0.014415    2.66   0.0166 Value of Outstanding Shares
                                                     Lagged, GE
ge_c           1  0.139036 0.024986    5.56   <.0001 Capital Stock Lagged, GE
```

```
                        The SYSLIN Procedure
              Seemingly Unrelated Regression Estimation

              Model                                WESTING
              Dependent Variable                      wh_i
              Label                   Gross Investment, WH


                         Parameter Estimates

            Parameter Standard                  Variable
Variable      DF  Estimate    Error t Value Pr > |t| Label

Intercept      1  -1.25199 7.545217   -0.17   0.8702 Intercept
wh_f           1  0.057630 0.014546    3.96   0.0010 Value of Outstanding Shares
                                                     Lagged, WH
wh_c           1  0.063978 0.053041    1.21   0.2443 Capital Stock Lagged, WH
```

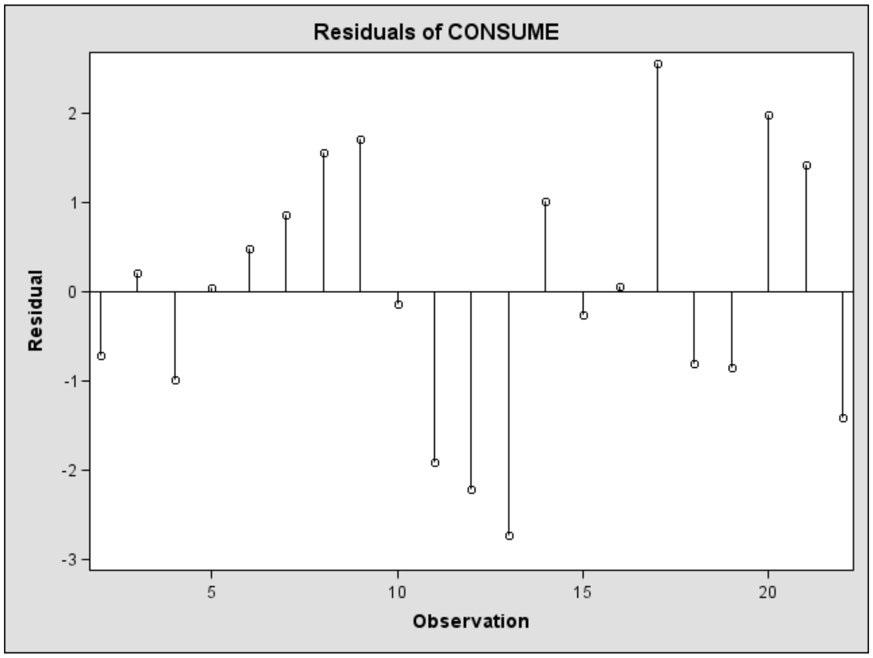
Example 26.3. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics. This is a continuation of Example 26.1 on page 1515. These graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the SYSLIN procedure, see the "ODS Graphics" section on page 1514.

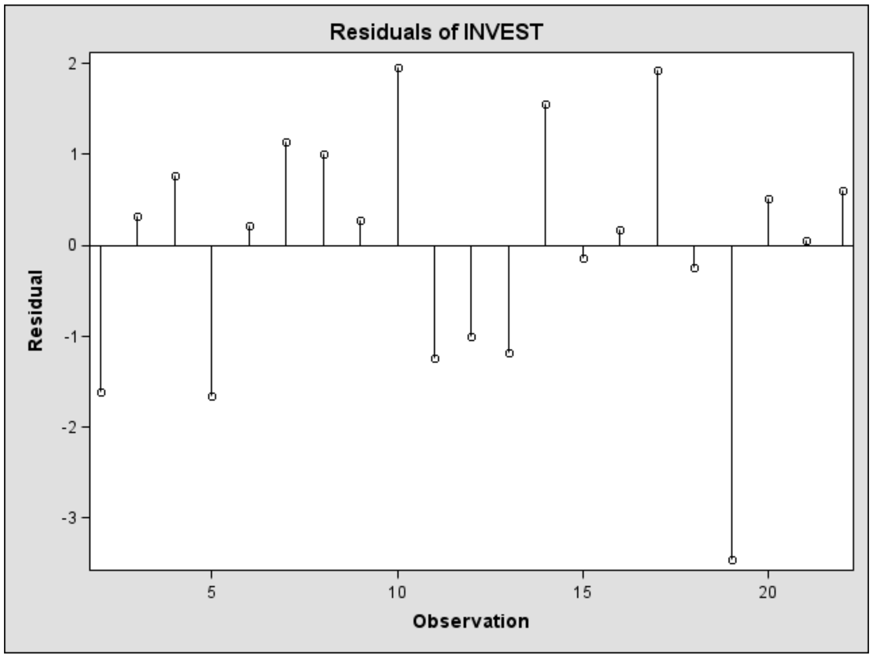
The following statements show how to generate ODS graphics plots with the SYSLIN procedure. The plots of residuals for each one of the equations in the model are displayed in Output 26.3.1 through Output 26.3.3.

```
ods html;
ods graphics on;

proc syslin data=klein outest=b liml;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model c = p plag  wsum;
   invest:  model i = p plag  klag;
   labor:   model w = x xlag  yr;
run;

ods graphics off;
ods html close;
```

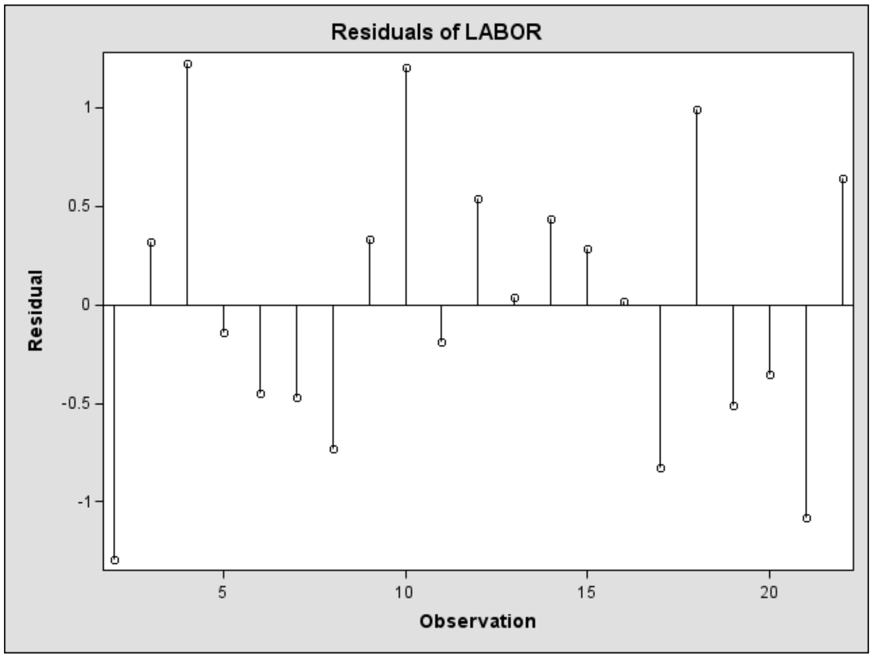
Output 26.3.1. Residuals Plot for Consumption (Experimental)



Output 26.3.2. Residuals Plot for Investments (Experimental)



Output 26.3.3. Residuals Plot for Labor (Experimental)



References

Basmann, R.L. (1960), "On Finite Sample Distributions of Generalized Classical Linear Identifiability Test Statistics," *Journal of the American Statistical Association*, 55, 650-659.

Fuller, W.A. (1977), "Some Properties of a Modification of the Limited Information Estimator," *Econometrica*, 45, 939-952.

Hausman, J.A. (1975), "An Instrumental Variable Approach to Full Information Estimators for Linear and Certain Nonlinear Econometric Models," *Econometrica*, 43, 727-738.

Johnston, J. (1984), *Econometric Methods*, Third Edition, New York: McGraw-Hill Book Company.

Judge, George G., W. E. Griffiths, R. Carter Hill, Helmut Lutkepohl, and Tsoung-Chao Lee (1985), *The Theory and Practice of Econometrics*, Second Edition, New York: John Wiley & Sons, Inc.

Maddala, G.S. (1977), *Econometrics*, New York: McGraw-Hill Book Company.

Park, S.B. (1982), "Some Sampling Properties of Minimum Expected Loss (MELO) Estimators of Structural Coefficients," *Journal of the Econometrics*, 18, 295-311.

Pindyck, R.S. and Rubinfeld, D.L. (1981), *Econometric Models and Economic Forecasts*, Second Edition, New York: McGraw-Hill Book Company.

Pringle, R.M. and Raynor, A.A. (1971), *Generalized Inverse Matrices with Applications to Statistics*, New York: Hafner Publishing Company.

Rao, P. (1974), “Specification Bias in Seemingly Unrelated Regressions,” in *Essays in Honor of Tinbergen*, Volume 2, New York: International Arts and Sciences Press.

Savin, N.E. and White, K.J. (1978), “Testing for Autocorrelation with Missing Observations,” *Econometrics*, 46, 59-66.

Theil, H. (1971), *Principles of Econometrics*, New York: John Wiley & Sons, Inc.

Zellner, A. (1962), “An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias,” *Journal of the American Statistical Association*, 57, 348-368.

Zellner, A. (1978), “Estimation of Functions of Population Means and Regression Coefficients: A Minimum Expected Loss (MELO) Approach,” *Journal of the Econometrics,* 8, 127-158.

Zellner, A. and Park, S. (1979), “Minimum Expected Loss (MELO) Estimators for Functions of Parameters and Structural Coefficients of Econometric Models,” *Journal of the American Statistical Association*, 74, 185-193.

Chapter 27
The TSCSREG Procedure

Chapter Contents

Chapter 27
The TSCSREG Procedure

Overview

The TSCSREG (**T**ime **S**eries **C**ross **S**ection **Reg**ression) procedure analyzes a class of linear econometric models that commonly arise when time series and cross-sectional data are combined. The TSCSREG procedure deals with panel data sets that consist of time series observations on each of several cross-sectional units.

Such models can be viewed as two-way designs with covariates

$$y_{it} = \sum_{k=1}^{K} X_{itk}\beta_k + u_{it} \quad i = 1, \ldots, N; \quad t = 1, \ldots, T$$

where *N* is the number of cross sections, *T* is the length of the time series for each cross section, and *K* is the number of exogenous or independent variables.

The performance of any estimation procedure for the model regression parameters depends on the statistical characteristics of the error components in the model. The TSCSREG procedure estimates the regression parameters in the preceding model under several common error structures. The error structures and the corresponding methods the TSCSREG procedure uses to analyze them are as follows:

- one and two-way fixed and random effects models. If the specification is dependent only on the cross section to which the observation belongs, such a model is referred to as a model with one-way effects. A specification that depends on both the cross section and the time series to which the observation belongs is called a model with two-way effects.
- Therefore, the specifications for the one-way model are

  $$u_{it} = \nu_i + \epsilon_{it}$$

  and the specifications for the two-way model are

  $$u_{it} = \nu_i + e_t + \epsilon_{it}$$

  where ϵ_{it} is a classical error term with zero mean and a homoscedastic covariance matrix.
- Apart from the possible one-way or two-way nature of the effect, the other dimension of difference between the possible specifications is that of the nature of the cross-sectional or time-series effect. The models are referred to as fixed effects models if the effects are nonrandom and as random effects models otherwise.

- first-order autoregressive model with contemporaneous correlation

$$u_{it} = \rho_i u_{i,t-1} + \epsilon_{it}$$

- The Parks method is used to estimate this model. This model assumes a first-order autoregressive error structure with contemporaneous correlation between cross sections. The covariance matrix is estimated by a two-stage procedure leading to the estimation of model regression parameters by GLS.
- mixed variance-component moving average error process

$$u_{it} = a_i + b_t + e_{it}$$

$$e_{it} = \alpha_0 \epsilon_t + \alpha_1 \epsilon_{t-1} + \ldots + \alpha_m \epsilon_{t-m}$$

- The Da Silva method is used to estimate this model. The Da Silva method estimates the regression parameters using a two-step GLS-type estimator.

The TSCSREG procedure analyzes panel data sets that consist of multiple time series observations on each of several individuals or cross-sectional units. The input data set must be in time series cross-sectional form. See Chapter 2, "Working with Time Series Data," for a discussion of how time series related by a cross-sectional dimension are stored in SAS data sets. The TSCSREG procedure requires that the time series for each cross section have the same number of observations and cover the same time range.

Getting Started

Specifying the Input Data

The input data set used by the TSCSREG procedure must be sorted by cross section and by time within each cross section. Therefore, the first step in using PROC TSCSREG is to make sure that the input data set is sorted. Normally, the input data set contains a variable that identifies the cross section for each observation and a variable that identifies the time period for each observation.

To illustrate, suppose that you have a data set A containing data over time for each of several states. You want to regress the variable Y on regressors X1 and X2. Cross sections are identified by the variable STATE, and time periods are identified by the variable DATE. The following statements sort the data set A appropriately:

```
proc sort data=a;
   by state date;
run;
```

The next step is to invoke the TSCSREG procedure and specify the cross section and time series variables in an ID statement. List the variables in the ID statement exactly as they are listed in the BY statement.

```
proc tscsreg data=a;
   id state date;
```

Alternatively, you can omit the ID statement and use the CS= and TS= options on the PROC TSCSREG statement to specify the number of cross sections in the data set and the number of time series observations in each cross section.

Unbalanced Data

In the case of fixed effects and random effects models, the TSCSREG procedure is capable of processing data with different numbers of time series observations across different cross sections. You must specify the ID statement to estimate models using unbalanced data. The missing time series observations are recognized by the absence of time series id variable values in some of the cross sections in the input data set. Moreover, if an observation with a particular time series id value and cross-sectional id value is present in the input data set, but one or more of the model variables are missing, that time series point is treated as missing for that cross section.

Also, when PROC TSCSREG is processing balanced data, you now need to specify only the CS= parameter if you do not specify an ID statement. The TS= parameter is not required, since it can be inferred from the number of observations if the data is balanced.

Specifying the Regression Model

Next, specify the linear regression model with a MODEL statement. The MODEL statement in PROC TSCSREG is specified like the MODEL statement in other SAS regression procedures: the dependent variable is listed first, followed by an equal sign, followed by the list of regressor variables.

```
proc tscsreg data=a;
   id state date;
   model y = x1 x2;
run;
```

The reason for using PROC TSCSREG instead of other SAS regression procedures is that you can incorporate a model for the structure of the random errors. It is important to consider what kind of error structure model is appropriate for your data and to specify the corresponding option in the MODEL statement.

The error structure options supported by the TSCSREG procedure are FIXONE, FIXTWO, RANONE, RANTWO, FULLER, PARKS, and DASILVA. See the "Details" section later in this chapter for more information about these methods and the error structures they assume.

By default, the Fuller-Battese method is used. Thus, the preceding example is the same as specifying the FULLER option, as shown in the following statements:

```
proc tscsreg data=a;
   id state date;
```

```
      model y = x1 x2 / fuller;
   run;
```

You can specify more than one error structure option in the MODEL statement; the analysis is repeated using each method specified. You can use any number of MODEL statements to estimate different regression models or estimate the same model using different options. See Example 27.1 in the section "Examples."

In order to aid in model specification within this class of models, the procedure provides two specification test statistics. The first is an *F* statistic that tests the null hypothesis that the fixed effects parameters are all zero. The second is a Hausman *m*-statistic that provides information about the appropriateness of the random effects specification. It is based on the idea that, under the null hypothesis of no correlation between the effects variables and the regressors, OLS and GLS are consistent, but OLS is inefficient. Hence, a test can be based on the result that the covariance of an efficient estimator with its difference from an inefficient estimator is zero. Rejection of the null hypothesis might suggest that the fixed effects model is more appropriate.

The procedure also provides the Buse R-squared measure, which is the most appropriate goodness-of-fit measure for models estimated using GLS. This number is interpreted as a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables. In the case of OLS estimation, the Buse R-squared measure is equivalent to the usual R-squared measure.

Estimation Techniques

If the effects are fixed, the models are essentially regression models with dummy variables corresponding to the specified effects. For fixed effects models, ordinary least squares (OLS) estimation is best linear unbiased.

The other alternative is to assume that the effects are random. In the one-way case, $E(\nu_i) = 0$, $E(\nu_i^2) = \sigma_\nu^2$, and

$E(\nu_i \nu_j) = 0$ for $i \neq j$, and ν_i is uncorrelated with ϵ_{it} for all i and t. In the two-way case, in addition to all of the preceding, $E(e_t) = 0$, $E(e_t^2) = \sigma_e^2$, and

$E(e_t e_s) = 0$ for $t \neq s$, and the e_t are uncorrelated with the ν_i and the ϵ_{it} for all $i$and t. Thus, the model is a variance components model, with the variance components σ_ν^2 and σ_e^2, as well as σ_ϵ^2, to be estimated. A crucial implication of such a specification is that the effects are independent of the regressors. For random effects models, the estimation method is an estimated generalized least squares (EGLS) procedure that involves estimating the variance components in the first stage and using the estimated variance covariance matrix thus obtained to apply generalized least squares (GLS) to the data.

Introductory Example

The following example uses the cost function data from Greene (1990) to estimate the variance components model. The variable OUTPUT is the log of output in millions of kilowatt-hours, and COST is the log of cost in millions of dollars. Refer to Greene (1990) for details.

```
data greene;
   input firm year output cost @@;
cards;
     1 1955   5.36598   1.14867  1 1960   6.03787   1.45185
     1 1965   6.37673   1.52257  1 1970   6.93245   1.76627
     2 1955   6.54535   1.35041  2 1960   6.69827   1.71109
     2 1965   7.40245   2.09519  2 1970   7.82644   2.39480
     3 1955   8.07153   2.94628  3 1960   8.47679   3.25967
     3 1965   8.66923   3.47952  3 1970   9.13508   3.71795
     4 1955   8.64259   3.56187  4 1960   8.93748   3.93400
     4 1965   9.23073   4.11161  4 1970   9.52530   4.35523
     5 1955   8.69951   3.50116  5 1960   9.01457   3.68998
     5 1965   9.04594   3.76410  5 1970   9.21074   4.05573
     6 1955   9.37552   4.29114  6 1960   9.65188   4.59356
     6 1965  10.21163   4.93361  6 1970  10.34039   5.25520
;

proc sort data=greene;
   by firm year;
run;
```

Usually you cannot explicitly specify all the explanatory variables that affect the dependent variable. The omitted or unobservable variables are summarized in the error disturbances. The TSCSREG procedure used with the Fuller-Battese method adds the individual and time-specific random effects to the error disturbances, and the parameters are efficiently estimated using the GLS method. The variance components model used by the Fuller-Battese method is

$$y_{it} = \sum_{k=1}^{K} X_{itk}\beta_k + v_i + e_t + \epsilon_{it} \quad i = 1, \ldots, N; \quad t = 1, \ldots, T$$

The following statements fit this model. Since the Fuller-Battese is the default method, no options are required.

```
proc tscsreg data=greene;
   model cost = output;
   id firm year;
run;
```

The TSCSREG procedure output is shown in Figure 27.1. A model description is printed first, which reports the estimation method used and the number of cross sections and time periods. The variance components estimates are printed next. Finally, the table of regression parameter estimates shows the estimates, standard errors, and *t*-tests.

```
                              The TSCSREG Procedure


Dependent Variable: cost

                               Model Description

                      Estimation Method             RanTwo
                      Number of Cross Sections           6
                      Time Series Length                 4


                                Fit Statistics

               SSE                0.3481    DFE                     22
               MSE                0.0158    Root MSE            0.1258
               R-Square           0.8136


                        Variance Component Estimates

               Variance Component for Cross Sections    0.046907
               Variance Component for Time Series        0.00906
               Variance Component for Error             0.008749


                               Hausman Test for
                                Random Effects

                              DF    m Value    Pr > m

                               1      26.46    <.0001


                              Parameter Estimates

                                                   Standard
       Variable         DF     Estimate         Error    t Value    Pr > |t|

       Intercept         1     -2.99992        0.6478      -4.63      0.0001
       output            1     0.746596        0.0762       9.80      <.0001
```

Figure 27.1. The Variance Components Estimates

Syntax

The following statements are used with the TSCSREG procedure.

PROC TSCSREG *options*;
  BY *variables*;
  ID *cross-section-id-variable time-series-id-variable*;
  MODEL *dependent = regressor-variables / options*;
  label: **TEST** *equation [,equation...]*;

Functional Summary

The statements and options used with the TSCSREG procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set	TSCSREG	DATA=
write parameter estimates to an output data set	TSCSREG	OUTEST=
include correlations in the OUTEST= data set	TSCSREG	CORROUT
include covariances in the OUTEST= data set	TSCSREG	COVOUT
specify number of time series observations	TSCSREG	TS=
specify number of cross sections	TSCSREG	CS=
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the cross section and time ID variables	ID	
Printing Control Options		
print correlations of the estimates	MODEL	CORRB
print covariances of the estimates	MODEL	COVB
suppress printed output	MODEL	NOPRINT
perform tests of linear hypotheses	TEST	
Model Estimation Options		
specify the one-way fixed effects model	MODEL	FIXONE
specify the two-way fixed effects model	MODEL	FIXTWO
specify the one-way random effects model	MODEL	RANONE
specify the one-way random effects model	MODEL	RANTWO
specify Fuller-Battese method	MODEL	FULLER
specify PARKS	MODEL	PARKS
specify Da Silva method	MODEL	DASILVA
specify order of the moving average error process for Da Silva method	MODEL	M=
print Φ matrix for Parks method	MODEL	PHI
print autocorrelation coefficients for Parks method	MODEL	RHO
suppress the intercept term	MODEL	NOINT
control check for singularity	MODEL	SINGULAR=

PROC TSCSREG Statement

PROC TSCSREG *options;*

The following options can be specified on the PROC TSCSREG statement.

DATA= *SAS-data-set*
names the input data set. The input data set must be sorted by cross section and by time period within cross section. If you omit DATA=, the most recently created SAS data set is used.

TS= *number*
specifies the number of observations in the time series for each cross section. The TS= option value must be greater than 1. The TS= option is required unless an ID statement is used. Note that the number of observations for each time series must be the same for each cross section and must cover the same time period.

CS= *number*
specifies the number of cross sections. The CS= option value must be greater than 1. The CS= option is required unless an ID statement is used.

OUTEST= *SAS-data-set*
names an output data set to contain the parameter estimates. When the OUTEST= option is not specified, the OUTEST= data set is not created. See the section "OUTEST= Data Set" later in this chapter for details on the structure of the OUTEST= data set.

OUTCOV
COVOUT
writes the covariance matrix of the parameter estimates to the OUTEST= data set. See the section "OUTEST= Data Set" later in this chapter for details.

OUTCORR
CORROUT
writes the correlation matrix of the parameter estimates to the OUTEST= data set. See the section "OUTEST= Data Set" later in this chapter for details.

In addition, any of the following MODEL statement options can be specified in the PROC TSCSREG statement: CORRB, COVB, FIXONE, FIXTWO, RANONE, RANTWO, FULLER, PARKS, DASILVA, NOINT, NOPRINT, M=, PHI, RHO, and SINGULAR=. When specified in the PROC TSCSREG statement, these options are equivalent to specifying the options for every MODEL statement. See the section "MODEL Statement" for a complete description of each of these options.

BY Statement

BY *variables ;*

A BY statement can be used with PROC TSCSREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set must be sorted by the BY variables as well as by cross section and time period within the BY groups.

When both an ID statement and a BY statement are specified, the input data set must be sorted first with respect to BY variables and then with respect to the cross section and time series ID variables. For example,

```
proc sort data=a;
   by byvar1 byvar2 csid tsid;
run;

proc tscsreg data=a;
   by byvar1 byvar2;
   id csid tsid;
   ...
run;
```

When both a BY statement and an ID statement are used, the data set may have a different number of cross sections or a different number of time periods in each BY group. If no ID statement is used, the CS=*N* and TS=*T* options must be specified and each BY group must contain $N \times T$ observations.

ID Statement

ID *cross-section-id-variable time-series-id-variable;*

The ID statement is used to specify variables in the input data set that identify the cross section and time period for each observation.

When an ID statement is used, the TSCSREG procedure verifies that the input data set is sorted by the cross section ID variable and by the time series ID variable within each cross section. The TSCSREG procedure also verifies that the time series ID values are the same for all cross sections.

To make sure the input data set is correctly sorted, use PROC SORT with a BY statement with the variables listed exactly as they are listed in the ID statement to sort the input data set.

```
proc sort data=a;
   by csid tsid;
run;

proc tscsreg data=a;
   id csid tsid;
   ... etc. ...
run;
```

If the ID statement is not used, the TS= and CS= options must be specified on the PROC TSCSREG statement. Note that the input data must be sorted by time within cross section, regardless of whether the cross section structure is given by an ID statement or by the options TS= and CS=.

If an ID statement is specified, the time series length *T* is set to the minimum number of observations for any cross section, and only the first *T* observations in each cross section are used. If both the ID statement and the TS= and CS= options are specified, the TS= and CS= options are ignored.

MODEL Statement

MODEL *response = regressors / options;*

The MODEL statement specifies the regression model and the error structure assumed for the regression residuals. The response variable on the left side of the equal sign is regressed on the independent variables listed after the equal sign. Any number of MODEL statements can be used. For each model statement only one response variable can be specified on the left side of the equal sign.

The error structure is specified by the FULLER, PARKS, and DASILVA options. More than one of these three options can be used, in which case the analysis is repeated for each error structure model specified.

Models can be given labels. Model labels are used in the printed output to identify the results for different models. If no label is specified, the response variable name is used as the label for the model. The model label is specified as follows:

label **: MODEL** . . . **;**

The following options can be specified on the MODEL statement after a slash (/).

CORRB
CORR
prints the matrix of estimated correlations between the parameter estimates.

COVB
VAR
prints the matrix of estimated covariances between the parameter estimates.

FIXONE
specifies that a one-way fixed effects model be estimated.

FIXTWO
specifies that a two-way fixed effects model be estimated.

RANONE
specifies that a one-way random effects model be estimated.

RANTWO
specifies that a two-way random effects model be estimated.

FULLER
specifies that the model be estimated using the Fuller-Battese method, which assumes a variance components model for the error structure. See "Fuller-Battese Method" later in this chapter for details. FULLER is the default.

PARKS
specifies that the model be estimated using the Parks method, which assumes a first-order autoregressive model for the error structure. See "Parks Method" later in this chapter for details.

DASILVA

specifies that the model be estimated using the Da Silva method, which assumes a mixed variance-component moving average model for the error structure. See "Da Silva Method" later in this chapter for details.

M= *number*

specifies the order of the moving average process in the Da Silva method. The M= value must be less than $T - 1$. The default is M=1.

PHI

prints the Φ matrix of estimated covariances of the observations for the Parks method. The PHI option is relevant only when the PARKS option is used. See "Parks Method" later in this chapter for details.

RHO

prints the estimated autocorrelation coefficients for the Parks method.

NOINT
NOMEAN

suppresses the intercept parameter from the model.

NOPRINT

suppresses the normal printed output.

SINGULAR= *number*

specifies a singularity criterion for the inversion of the matrix. The default depends on the precision of the computer system.

TEST Statement

label: **TEST** *equation [,equation...]***;**

The TEST statement performs *F*-tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT refers to the coefficient of the intercept.

The following illustrates the use of the TEST statement:

```
proc tscsreg;
   model y = x1 x2 x3;
   test x1 = 0, x2/2 + 2*x3= 0;
   test_int: test intercept=0, x3 = 0;
```

Details

Notation

The discussion here is in the context of the usual panel structure,

$$y_{it} = \sum_{k=1}^{K} x_{itk}\beta_k + u_{it} \quad i = 1, \ldots N; \quad t = 1, \ldots T_i$$

with the specification of u_{it} dependent on the particular model. The total number of observations $M = \sum_{i=1}^{N} T_i$. For the balanced data case, $T_i = T$ for all i. The $M \times M$ covariance matrix of u_{it} is denoted by $\mathbf{V}$. Let $\mathbf{X}$ and $\mathbf{y}$ be the independent and dependent variables arranged by cross section and by time within each cross section. Let $\mathbf{X}_s$ be the X matrix without the intercept. Generally, all other notation is specific to each section.

The One-Way Fixed Effects Model

The specification for the one-way fixed effects model is

$$u_{it} = \nu_i + \epsilon_{it}$$

where the ν_is are nonrandom. Since including both the intercept and all the ν_is induces a redundancy (unless the intercept is suppressed with the NOINT option), the ν_i estimates are reported under the restriction that $\nu_N = 0$.

Let $\mathbf{Q}_0 = diag(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$.

The estimators for the intercept and the fixed effects are given by the usual OLS expressions.

If $\tilde{\mathbf{X}}_s = \mathbf{Q}_0\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q}_0\mathbf{y}$, the estimator of the slope coefficients is given by

$$\tilde{\beta}_s = (\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}$$

The estimator of the error variance is

$$\hat{\sigma}_\epsilon = \tilde{\mathbf{u}}'\mathbf{Q}_0\tilde{\mathbf{u}}/(M - N - (K-1))$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{J}_M\mathbf{j}'_M/M)(\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$if there is not.

The Two-Way Fixed Effects Model

The specification for the two-way fixed effects model is

$$u_{it} = \nu_i + e_t + \epsilon_{it}$$

where the ν_is and e_ts are nonrandom. If you do not specify the NOINT option, which suppresses the intercept, the estimates for the fixed effects are reported under the restriction that $\nu_N = 0$ and $e_T = 0$. If you specify the NOINT option to suppress the intercept, only the restriction $e_T = 0$ is imposed.

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables arranged by time and by cross section within each time period. (Note that the input data set used by the TSCSREG procedure must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections observed in year t and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix obtained from the $N \times N$ identity matrix from which rows corresponding to cross sections not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1^{'}, \mathbf{D}_2^{'}, \ldots..\mathbf{D}_T^{'})^{'}$and $\mathbf{Z}_2 = diag(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots\ldots\mathbf{D}_T\mathbf{j}_N)$. The matrix $\mathbf{Z}$ gives the dummy variable structure for the two-way model.

Let

$$\Delta_N = \mathbf{Z}_1^{'}\mathbf{Z}_1, \quad \Delta_T = \mathbf{Z}_2^{'}\mathbf{Z}_2, \quad \mathbf{A} = \mathbf{Z}_2^{'}\mathbf{Z}_1$$

$$\bar{\mathbf{Z}} = \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}^{'}$$

$$\mathbf{Q} = \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}^{'}$$

$$\mathbf{P} = (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}) - \bar{\mathbf{Z}}\mathbf{Q}^{-}\bar{\mathbf{Z}}^{'}$$

The estimators for the intercept and the fixed effects are given by the usual OLS expressions.

The estimate of the regression slope coefficients is given by

$$\tilde{\beta}_s = (\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{y}_*$$

where $\mathbf{X}_{*s}$ is the $\mathbf{X}_*$ matrix without the vector of 1s.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}^{'}\mathbf{P}\tilde{\mathbf{u}}/(M - T - N + 1 - (K - 1))$$

where the residuals are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M\mathbf{j}_M^{'}/M)(\mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s)$if there is an intercept in the model and by $\tilde{\mathbf{u}} = \mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s$if there is no intercept.

The One-Way Random Effects Model

The specification for the one-way random effects model is

$$u_{it} = \nu_i + \epsilon_{it}$$

Let $\mathbf{Z}_0 = diag(\mathbf{j}_{T_i})$, $\mathbf{P}_0 = diag(\bar{\mathbf{J}}_{T_i})$, and $\mathbf{Q}_0 = diag(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$. Define $\tilde{\mathbf{X}}_s = \mathbf{Q}_0\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q}_0\mathbf{y}$.

The fixed effects estimator of σ_ϵ^2 is still unbiased under the random effects assumptions, so you need to calculate only the estimate of σ_ν.

In the balanced data case, the estimation method for the variance components is the fitting constants method as applied to the one way model; refer to Baltagi and Chang (1994). Fuller and Battese (1974) apply this method to the two-way model.

Let

$$R(\nu) = \mathbf{y}^{'}\mathbf{Z}_0(\mathbf{Z}_0^{'}\mathbf{Z}_0)^{-1}\mathbf{Z}_0^{'}\mathbf{y}$$

$$R(\beta|\nu) = ((\tilde{\mathbf{X}}_s^{'}\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s^{'}\tilde{\mathbf{y}})^{'}(\tilde{\mathbf{X}}_s^{'}\tilde{\mathbf{y}})$$

$$R(\beta) = (\mathbf{X}^{'}\mathbf{y})^{'}(\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{y}$$

$$R(\nu|\beta) = R(\beta|\nu) + R(\nu) - R(\beta)$$

The estimator of the error variance is given by

$$\hat{\sigma}_\epsilon^2 = (\mathbf{y}^{'}\mathbf{y} - R(\beta|\nu) - R(\nu))/(M - N - (K-1))$$

and the estimator of the cross-sectional variance component is given by

$$\hat{\sigma}_\nu^2 = (R(\nu|\beta) - (N-1)\hat{\sigma}_\epsilon^2)/(M - \mathrm{tr}(\mathbf{Z}_0^{'}\mathbf{X}(\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{Z}_0))$$

The estimation of the one-way unbalanced data model is performed using a specialization (Baltagi and Chang 1994) of the approach used by Wansbeek and Kapteyn (1989) for unbalanced two-way models.

The estimation of the variance components is performed by using a quadratic unbiased estimation (QUE) method. This involves focusing on quadratic forms of the centered residuals, equating their expected values to the realized quadratic forms, and solving for the variance components.

Let

$$q_1 = \tilde{\mathbf{u}}^{'}\mathbf{Q}_0\tilde{\mathbf{u}}$$

$$q_2 = \tilde{\mathbf{u}}^{'}\mathbf{P}_0\tilde{\mathbf{u}}$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M\mathbf{j}'_M/M)(\mathbf{y} - \mathbf{X}_s\tilde{\mathbf{X}}'_s\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}})$ if there is an intercept and by $tilde\mathbf{u} = (\mathbf{y} - \mathbf{X}_s(\tilde{\mathbf{X}}'_s\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}'_s\tilde{\mathbf{y}})$ if there is not.

Consider the expected values

$$E(q_1) = (M - N - (K-1))\sigma_\epsilon^2$$

$$E(q_2) = (N - 1 + \mathrm{tr}[(\mathbf{X}_\mathrm{s}^{'}\mathbf{Q}_0\mathbf{X}_\mathrm{s})^{-1}\mathbf{X}_\mathrm{s}^{'}\mathbf{P}_0\mathbf{X}_\mathrm{s}] - \mathrm{tr}[(\mathbf{X}_\mathrm{s}^{'}\mathbf{Q}_0\mathbf{X}_\mathrm{s})^{-1}\mathbf{X}_\mathrm{s}^{'}\bar{\mathbf{J}}_\mathrm{M}\mathbf{X}_\mathrm{s}])\sigma_\epsilon^2$$

$$+[M - (\sum_i T_i^2/M)]\sigma_\nu^2$$

$\hat{\sigma}_\epsilon^2$ and $\hat{\sigma}_\nu^2$ are obtained by equating the quadratic forms to their expected values.

The estimated generalized least squares procedure substitutes the QUE estimates into the covariance matrix of u_{it}, which is given by

$$\mathbf{V} = \sigma_\nu^2 I_M + \sigma_\epsilon^2\mathbf{Z}_0\mathbf{Z}_0^{'}$$

The Two-Way Random Effects Model

The specification for the two way model is

$$u_{it} = \nu_i + e_t + \epsilon_{it}$$

For balanced data, the two-way random effects model is estimated using the method of Fuller and Battese (1974), so in this case, the RANTWO option is equivalent to the FULLER option already existing in PROC TSCSREG.

The following method (Wansbeek and Kapteyn 1989) is used to handle unbalanced data.

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables arranged by time and by cross section within each time period. (Note that the input data set used by the TSCSREG procedure must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections observed in time t and $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix obtained from the $N \times N$ identity matrix from which rows corresponding to cross sections not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1^{'}, \mathbf{D}_2^{'}, \ldots..\mathbf{D}_T^{'})^{'}$and $\mathbf{Z}_2 = diag(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots\ldots\mathbf{D}_T\mathbf{j}_N)$.

The matrix $\mathbf{Z}$ gives the dummy variable structure for the two-way model.

Let

$$\Delta_N = \mathbf{Z}_1^{'}\mathbf{Z}_1, \quad \Delta_T = \mathbf{Z}_2^{'}\mathbf{Z}_2, \quad \mathbf{A} = \mathbf{Z}_2^{'}\mathbf{Z}_1$$

$$\bar{\mathbf{Z}} = \mathbf{Z}_2 - \mathbf{Z}_1 \Delta_N^{-1} \mathbf{A}'$$

$$\mathbf{Q} = \Delta_T - \mathbf{A} \Delta_N^{-1} \mathbf{A}'$$

$$\mathbf{P} = (\mathbf{I}_M - \mathbf{Z}_1 \Delta_N^{-1} \mathbf{Z}_1') - \bar{\mathbf{Z}} \mathbf{Q} - \bar{\mathbf{Z}}'$$

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}' \mathbf{P} \tilde{\mathbf{u}} / M - T - N + 1 - (K - 1))$$

where the $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M \mathbf{j'}_M / M)(\mathbf{y}_* - \mathbf{X}_{*s} (\mathbf{X'}_{*s} \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X}_{*s}' \mathbf{P} \mathbf{y}_*)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y}_* - \mathbf{X}_{*s} (\mathbf{X'}_{*s} \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X'}_{*s} \mathbf{P} \mathbf{y}_*$if there is not.

The estimation of the variance components is performed by using a quadratic unbiased estimation (QUE) method that involves focusing on quadratic forms of the residuals $\tilde{\mathbf{u}}$, equating their expected values to the realized quadratic forms, and solving for the variance components.

Let

$$q_N = \tilde{\mathbf{u}}' \mathbf{Z_2} \Delta_T^{-1} \mathbf{Z}_2' \tilde{\mathbf{u}}$$

$$q_T = \tilde{\mathbf{u}}' \mathbf{Z}_1 \Delta_N^{-1} \mathbf{Z}_1' \tilde{\mathbf{u}}$$

Consider the expected values

$$E(q_N) = (T + k_N - (1 + k_0))\sigma^2 + (T - \frac{\lambda_1}{M})\sigma_\nu^2 + (M - \frac{\lambda_2}{M})\sigma_e^2$$

$$E(q_T) = (N + k_T - (1 + k_0))\sigma^2 + (M - \frac{\lambda_1}{M})\sigma_\nu^2 + (N - \frac{\lambda_2}{M})\sigma_e^2$$

where

$$k_0 = \mathbf{j}_M' \mathbf{X}_{*s} (\mathbf{X}_{*s}' \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X}_{*s}' \mathbf{j}_M / M$$

$$k_N = tr((\mathbf{X}_{*s}' \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X}_{*s}' \mathbf{Z}_2 \Delta_T^{-1} \mathbf{Z}_2' \mathbf{X}_{*s})$$

$$k_T = tr((\mathbf{X}_{*s}' \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X}_{*s}' \mathbf{Z}_1 \Delta_N^{-1} \mathbf{Z}_1' \mathbf{X}_{*s})$$

$$\lambda_1 = \mathbf{j}_M' \mathbf{Z}_1 \mathbf{Z}_1' \mathbf{j}_M$$

$$\lambda_2 = \mathbf{j}_M' \mathbf{Z}_2 \mathbf{Z}_2' \mathbf{j}_M$$

The quadratic unbiased estimators for σ_{ν}^{2} and σ_{e}^{2} are obtained by equating the expected values to the quadratic forms and solving for the two unknowns.

The estimated generalized least squares procedure substitute the QUE estimates into the covariance matrix of the composite error term u_{it}, which is given by

$$\mathbf{V} = \sigma_{\epsilon}^{2}\mathbf{I}_{M} + \sigma_{\nu}^{2}\mathbf{Z}_{1}\mathbf{Z}_{1}^{'} + \sigma_{e}^{2}\mathbf{Z}_{2}\mathbf{Z}_{2}^{'}$$

Parks Method (Autoregressive Model)

Parks (1967) considered the first-order autoregressive model in which the random errors $u_{it}, \quad i = 1, 2, \ldots, N, \quad t = 1, 2, \ldots, T$, have the structure

$$\begin{array}{rcll} E(u_{it}^{2}) & = & \sigma_{ii} & \text{(heteroscedasticity)} \\ E(u_{it}u_{jt}) & = & \sigma_{ij} & \text{(contemporaneously correlated)} \\ u_{it} & = & \rho_{i}u_{i,t-1} + \epsilon_{it} & \text{(autoregression)} \end{array}$$

where

$$\begin{array}{rcll} E(\epsilon_{it}) & = & 0 & \\ E(u_{i,t-1}\epsilon_{jt}) & = & 0 & \\ E(\epsilon_{it}\epsilon_{jt}) & = & \phi_{ij} & \\ E(\epsilon_{it}\epsilon_{js}) & = & 0 & (s{\neq}t) \\ E(u_{i0}) & = & 0 & \\ E(u_{i0}u_{j0}) & = & \sigma_{ij} = \phi_{ij}/(1-\rho_{i}\rho_{j}) & \end{array}$$

The model assumed is first-order autoregressive with contemporaneous correlation between cross sections. In this model, the covariance matrix for the vector of random errors **u** can be expressed as

$$E(\mathbf{u}\mathbf{u}^{'}) = \mathbf{V} = \begin{bmatrix} \sigma_{11}P_{11} & \sigma_{12}P_{12} & \ldots & \sigma_{1N}P_{1N} \\ \sigma_{21}P_{21} & \sigma_{22}P_{22} & \ldots & \sigma_{2N}P_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{N1}P_{N1} & \sigma_{N2}P_{N2} & \ldots & \sigma_{NN}P_{NN} \end{bmatrix}$$

where

$$P_{ij} = \begin{bmatrix} 1 & \rho_j & \rho_j^2 & \dots & \rho_j^{T-1} \\ \rho_i & 1 & \rho_j & \dots & \rho_j^{T-2} \\ \rho_i^2 & \rho_i & 1 & \dots & \rho_j^{T-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho_i^{T-1} & \rho_i^{T-2} & \rho_i^{T-3} & \dots & 1 \end{bmatrix}$$

The matrix **V** is estimated by a two-stage procedure, and β is then estimated by generalized least squares. The first step in estimating **V** involves the use of ordinary least squares to estimate β and obtain the fitted residuals, as follows:

$$\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\beta}_{OLS}$$

A consistent estimator of the first-order autoregressive parameter is then obtained in the usual manner, as follows:

$$\hat{\rho}_i = \left(\sum_{t=2}^{T} \hat{u}_{it}\hat{u}_{i,t-1} \right) \Bigg/ \left(\sum_{t=2}^{T} \hat{u}_{i,t-1}^2 \right) \quad i = 1, 2, \dots, N$$

Finally, the autoregressive characteristic of the data can be removed (asymptotically) by the usual transformation of taking weighted differences. That is, for $i = 1, 2, \dots, N$,

$$y_{i1}\sqrt{1-\hat{\rho}_i^2} = \sum_{k=1}^{p} X_{i1k}\beta_k\sqrt{1-\hat{\rho}_i^2} + u_{i1}\sqrt{1-\hat{\rho}_i^2}$$

$$y_{it} - \hat{\rho}_i y_{i,t-1} = \sum_{k=1}^{p} (X_{itk} - \hat{\rho}_i \mathbf{X}_{i,t-1,k})\beta_k + u_{it} - \hat{\rho}_i u_{i,t-1} \quad t = 2, \dots, T$$

which is written

$$y_{it}^* = \sum_{k=1}^{p} X_{itk}^*\beta_k + u_{it}^* \quad i = 1, 2, \dots, N; \quad t = 1, 2, \dots, T$$

Notice that the transformed model has not lost any observations (Seely and Zyskind 1971).

The second step in estimating the covariance matrix **V** is to apply ordinary least squares to the preceding transformed model, obtaining

$$\hat{\mathbf{u}}^* = \mathbf{y}^* - \mathbf{X}^* \beta^*_{OLS}$$

from which the consistent estimator of σ_{ij} is calculated:

$$s_{ij} = \frac{\hat{\phi}_{ij}}{(1 - \hat{\rho}_i \hat{\rho}_j)}$$

where

$$\hat{\phi}_{ij} = \frac{1}{(T-p)} \sum_{t=1}^{T} \hat{u}^*_{it} \hat{u}^*_{jt}$$

EGLS then proceeds in the usual manner,

$$\hat{\beta}_P = (\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{y}$$

where $\hat{\mathbf{V}}$ is the derived consistent estimator of **V**. For computational purposes, it should be pointed out that $\hat{\beta}_P$ is obtained directly from the transformed model,

$$\hat{\beta}_P = (\mathbf{X}^{*'}(\hat{\Phi}^{-1} \otimes I_T)\mathbf{X}^*)^{-1}\mathbf{X}^{*'}(\hat{\Phi}^{-1} \otimes I_T)\mathbf{y}^*$$

where $\hat{\Phi} = [\hat{\phi}_{ij}]_{i,j=1,\ldots,N}$.

The preceding procedure is equivalent to Zellner's two-stage methodology applied to the transformed model (Zellner 1962).

Parks demonstrates that his estimator is consistent and asymptotically, normally distributed with

$$\mathrm{Var}(\hat{\beta}_{\mathrm{P}}) = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$$

Standard Corrections

For the PARKS option, the first-order autocorrelation coefficient must be estimated for each cross section. Let ρ be the $N * 1$ vector of true parameters and $R = (r_1, \ldots, r_N)'$ be the corresponding vector of estimates. Then, to ensure that only range-preserving estimates are used in PROC TSCSREG, the following modification for R is made:

$$r_i = \begin{cases} r_i & \text{if } |r_i| < 1 \\ max(.95, rmax) & \text{if } r_i \geq 1 \\ min(-.95, rmin) & \text{if } r_i \leq -1 \end{cases}$$

where

$$rmax = \begin{cases} 0 & \text{if } r_i < 0 \text{ or } r_i \geq 1 \text{ for all i} \\ \max_j [r_j : 0 \leq r_j < 1] & \text{otherwise} \end{cases}$$

and

$$rmin = \begin{cases} 0 & \text{if } r_i > 0 \text{ or } r_i \leq -1 \text{ for all i} \\ \max_j [r_j : -1 < r_j \leq 0] & \text{otherwise} \end{cases}$$

Whenever this correction is made, a warning message is printed.

Da Silva Method (Variance-Component Moving Average Model)

Suppose you have a sample of observations at *T* time points on each of *N* cross-sectional units. The Da Silva method assumes that the observed value of the dependent variable at the *t*th time point on the *i*th cross-sectional unit can be expressed as

$$y_{it} = \mathbf{x}_{it}^{'}\beta + a_i + b_t + e_{it} \quad i = 1, \ldots, N; \quad t = 1, \ldots, T$$

where

$\mathbf{x}_{it}^{'} = (x_{it1}, \ldots, x_{itp})$is a vector of explanatory variables for the *t*th time point and *i*th cross-sectional unit

$\beta = (\beta_1, \ldots, \beta_p)'$ is the vector of parameters

a_i is a time-invariant, cross-sectional unit effect

b_t is a cross-sectionally invariant time effect

e_{it} is a residual effect unaccounted for by the explanatory variables and the specific time and cross-sectional unit effects

Since the observations are arranged first by cross sections, then by time periods within cross sections, these equations can be written in matrix notation as

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$$

where

$$\mathbf{u} = (\mathbf{a} \otimes \mathbf{1}_T) + (\mathbf{1}_N \otimes \mathbf{b}) + \mathbf{e}$$

$$\mathbf{y} = (y_{11}, \ldots, y_{1T}, y_{21}, \ldots, y_{NT})'$$

$$\mathbf{X} = (\mathbf{x}_{11}, \ldots, \mathbf{x}_{1T}, \mathbf{x}_{21}, \ldots, \mathbf{x}_{NT})'$$

$$\mathbf{a} = (a_1 \ldots a_N)'$$

$$\mathbf{b} = (b_1 \ldots b_T)'$$

$$\mathbf{e} = (e_{11}, \ldots, e_{1T}, e_{21}, \ldots, e_{NT})'$$

Here $\mathbf{1}_N$ is an $N \times 1$ vector with all elements equal to 1, and $\otimes$ denotes the Kronecker product.

It is assumed that

1. $\mathbf{x}_{it}$ is a sequence of nonstochastic, known $p \times 1$ vectors in $\Re^p$ whose elements are uniformly bounded in $\Re^p$. The matrix **X** has a full column rank *p*.
2. β is a $p \times 1$ constant vector of unknown parameters.
3. **a** is a vector of uncorrelated random variables such that $E(a_i) = 0$ and $var(a_i) = \sigma_a^2$, $\sigma_a^2 > 0, i = 1, \ldots, N$.
4. **b** is a vector of uncorrelated random variables such that $E(b_t) = 0$ and $var(b_t) = \sigma_b^2, \sigma_b^2 > 0, t = 1, \ldots, T$.
5. $\mathbf{e}_i = (e_{i1}, \ldots, e_{iT})'$ is a sample of a realization of a finite moving average time series of order $m < T - 1$ for each *i*; hence,

$$e_{it} = \alpha_0 \epsilon_t + \alpha_1 \epsilon_{t-1} + \ldots + \alpha_m \epsilon_{t-m}, \quad t = 1, \ldots, T; \quad i = 1, \ldots, N$$

   where $\alpha_0, \alpha_1, \ldots, \alpha_m$ are unknown constants such that $\alpha_0 \neq 0$ and $\alpha_m \neq 0$, and $\{\epsilon_j\}_{j=-\infty}^{j=\infty}$ is a white noise process, that is, a sequence of uncorrelated random variables with $E(\epsilon_t) = 0, E(\epsilon_t^2) = \sigma_\epsilon^2$, and $\sigma_\epsilon^2 > 0$.
6. The sets of random variables $\{a_i\}_{i=1}^N$, $\{b_t\}_{t=1}^T$, and $\{e_{it}\}_{t=1}^T$ for $i = 1, \ldots, N$ are mutually uncorrelated.
7. The random terms have normal distributions: $a_i \sim N(0, \sigma_a^2), b_t \sim N(0, \sigma_b^2)$,and $\epsilon_{t-k} \sim N(0, \sigma_\epsilon^2)$, for $i = 1, \ldots, N; t = 1, \ldots T; k = 1, \ldots, m$.

If assumptions 1-6 are satisfied, then

$$E(\mathbf{y}) = \mathbf{X}\beta$$

and

$$var(\mathbf{y}) = \sigma_a^2 (I_N \otimes J_T) + \sigma_b^2 (J_N \otimes I_T) + (I_N \otimes \Gamma_T)$$

where Γ_T is a $T \times T$ matrix with elements γ_{ts} as follows:

$$cov(e_{it}e_{is}) = \begin{cases} \gamma(|t-s|) & \text{if } |t-s| \leq \text{m} \\ 0 & \text{if } |t-s| > \text{m} \end{cases}$$

where $\gamma(k) = \sigma_\epsilon^2 \sum_{j=0}^{m-k} \alpha_j \alpha_{j+k}$ for $k = |t-s|$. For the definition of I_N, I_T, J_N, and J_T, see the "Fuller-Battese Method" section earlier in this chapter.

The covariance matrix, denoted by **V**, can be written in the form

$$\mathbf{V} = \sigma_a^2(I_N \otimes J_T) + \sigma_b^2(J_N \otimes I_T) + \sum_{k=0}^{m} \gamma(k)(I_N \otimes \Gamma_T^{(k)})$$

where $\Gamma_T^{(0)} = I_T$, and, for *k*=1,. . ., *m*, $\Gamma_T^{(k)}$ is a band matrix whose *k*th off-diagonal elements are 1's and all other elements are 0's.

Thus, the covariance matrix of the vector of observations **y** has the form

$$var(\mathbf{y}) = \sum_{k=1}^{m+3} \nu_k V_k$$

where

$$\begin{array}{lcll} \nu_1 & = & \sigma_a^2 & \\ \nu_2 & = & \sigma_b^2 & \\ \nu_k & = & \gamma(k-3) & k = 3, \ldots, m+3 \\ V_1 & = & I_N \otimes J_T & \\ V_2 & = & J_N \otimes I_T & \\ V_k & = & I_N \otimes \Gamma_T^{(k-3)} & k = 3, \ldots, m+3 \end{array}$$

The estimator of β is a two-step GLS-type estimator, that is, GLS with the unknown covariance matrix replaced by a suitable estimator of **V**. It is obtained by substituting Seely estimates for the scalar multiples $\nu_k, k = 1, 2, \ldots, m+3$.

Seely (1969) presents a general theory of unbiased estimation when the choice of estimators is restricted to finite dimensional vector spaces, with a special emphasis on quadratic estimation of functions of the form $\sum_{i=1}^{n} \delta_i \nu_i$.

The parameters ν_i (*i*=1,. . ., n) are associated with a linear model E(**y**)=**X**β with covariance matrix $\sum_{i=1}^{n} \nu_i V_i$ where V_i (*i*=1, . . ., n) are real symmetric matrices. The method is also discussed by Seely (1970a,1970b) and Seely and Zyskind (1971). Seely and Soong (1971) consider the MINQUE principle, using an approach along the lines of Seely (1969).

Linear Hypothesis Testing

For a linear hypothesis of the form $\mathbf{R}\ \beta$=$\mathbf{r}$ where $\mathbf{R}$ is $J{\times}L$ and $\mathbf{r}$ is $J{\times}1$, the *F*-statistic with $J, M-L$ degrees of freedom is computed as

$$(\mathbf{R}\beta-\mathbf{r})^{'}[\mathbf{R}(\mathbf{X}^{'}\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{R}']^{-1}\mathbf{R}(\mathbf{R}\beta-\mathbf{r})$$

R-squared

The conventional R-squared measure is inappropriate for all models that the TSCSREG procedure estimates using GLS since a number outside the 0-to-1 range may be produced. Hence, a generalization of the R-squared measure is reported. The following goodness-of-fit measure (Buse 1973) is reported:

$$R^2 = 1 - \frac{\hat{\mathbf{u}}^{'}\hat{\mathbf{V}}^{-1}\hat{\mathbf{u}}}{\mathbf{y}^{'}\mathbf{D}^{'}\hat{\mathbf{V}}^{-1}\mathbf{D}\mathbf{y}}$$

where $\hat{\mathbf{u}}$ are the residuals of the transformed model, $\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}(\mathbf{X}^{'}\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\hat{\mathbf{V}}^{-1}\mathbf{y}$,

and $\mathbf{D} = \mathbf{I}_M - \mathbf{j}_M\mathbf{j}_M^{'}(\frac{\hat{\mathbf{V}}^{-1}}{\mathbf{j}_M^{'}\hat{\mathbf{V}}^{-1}\mathbf{j}_M})$.

This is a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables.

If there is no intercept in the model, the corresponding measure (Theil 1961) is

$$R^2 = 1 - \frac{\hat{\mathbf{u}}^{'}\hat{\mathbf{V}}^{-1}\hat{\mathbf{u}}}{\mathbf{y}^{'}\hat{\mathbf{V}}^{-1}\mathbf{y}}$$

Clearly, in the case of OLS estimation, both the R-squared formulas given here reduce to the usual R-squared formula.

Specification Tests

The TSCSREG procedure outputs the results of one specification test for fixed effects and one specification test for random effects.

For fixed effects, let β_f be the n dimensional vector of fixed effects parameters. The specification test reported is the conventional *F*-statistic for the hypothesis $\beta_f = \mathbf{0}$. The *F*-statistic with $n, M-K$ degrees of freedom is computed as

$$\hat{\beta}_f\hat{\mathbf{S}}_f^{-1}\hat{\beta}_f/n$$

where $\hat{\mathbf{S}}_f$ is the estimated covariance matrix of the fixed effects parameters.

Hausman's (1978) specification test or *m*-statistic can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu

(1973) and further extended in Hausman and Taylor (1982). Hausman's *m*-statistic is as follows.

Consider two estimators, $\hat{\beta}_a$ and $\hat{\beta}_b$, which under the null hypothesis are both consistent, but only $\hat{\beta}_a$ is asymptotically efficient. Under the alternative hypothesis, only $\hat{\beta}_b$ is consistent. The *m*-statistic is

$$m = (\hat{\beta}_b - \hat{\beta}_a)'(\hat{\mathbf{S}}_b - \hat{\mathbf{S}}_a)^{-}(\hat{\beta}_b - \hat{\beta}_a)$$

where $\hat{\mathbf{S}}_b$ and $\hat{\mathbf{S}}_a$ are consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_b$ and $\hat{\beta}_a$. Then m is distributed χ^2 with k degrees of freedom, where k is the dimension of $\hat{\beta}_a$ and $\hat{\beta}_b$.

In the random effects specification, the null hypothesis of no correlation between effects and regressors implies that the OLS estimates of the slope parameters are consistent and inefficient but the GLS estimates of the slope parameters are consistent and efficient. This facilitates a Hausman specification test. The reported χ^2 statistic has degrees of freedom equal to the number of slope parameters.

OUTEST= Data Set

PROC TSCSREG writes the parameter estimates to an output data set when the OUTEST= option is specified. The OUTEST= data set contains the following variables:

MODEL	a character variable containing the label for the MODEL statement if a label is specified
METHOD	a character variable identifying the estimation method. Current methods are FULLER, PARKS, and DASILVA.
TYPE	a character variable that identifies the type of observation. Values of the _TYPE_ variable are CORRB, COVB, CSPARMS, and PARMS; the CORRB observation contains correlations of the parameter estimates; the COVB observation contains covariances of the parameter estimates; the CSPARMS observation contains cross-sectional parameter estimates; and the PARMS observation contains parameter estimates.
NAME	a character variable containing the name of a regressor variable for COVB and CORRB observations and left blank for other observations. The _NAME_ variable is used in conjunction with the _TYPE_ values COVB and CORRB to identify rows of the correlation or covariance matrix.
DEPVAR	a character variable containing the name of the response variable
MSE	the mean square error of the transformed model
CSID	the value of the cross section ID for CSPARMS observations. _CSID_ is used with the _TYPE_ value CSPARMS to identify the cross section for the first order autoregressive parameter estimate

	contained in the observation. _CSID_ is missing for observations with other _TYPE_ values. (Currently only the _A_1 variable contains values for CSPARMS observations.)
VARCS	the variance component estimate due to cross sections. _VARCS_ is included in the OUTEST= data set when either the FULLER or DASILVA option is specified.
VARTS	the variance component estimate due to time series. _VARTS_ is included in the OUTEST= data set when either the FULLER or DASILVA option is specified.
VARERR	the variance component estimate due to error. _VARERR_ is included in the OUTEST= data set when the FULLER option is specified.
_A_1	the first order autoregressive parameter estimate. _A_1 is included in the OUTEST= data set when the PARKS option is specified. The values of _A_1 are cross-sectional parameters, meaning that they are estimated for each cross section separately. _A_1 has a value only for _TYPE_=CSPARMS observations. The cross section to which the estimate belongs is indicated by the _CSID_ variable.
INTERCEP	the intercept parameter estimate. (INTERCEP will be missing for models for which the NOINT option is specified.)
regressors	the regressor variables specified in the MODEL statement. The regressor variables in the OUTEST= data set contain the corresponding parameter estimates for the model identified by _MODEL_ for _TYPE_=PARMS observations, and the corresponding covariance or correlation matrix elements for _TYPE_=COVB and _TYPE_=CORRB observations. The response variable contains the value -1 for the _TYPE_=PARMS observation for its model.

Printed Output

For each MODEL statement, the printed output from PROC TSCSREG includes the following:

1. a model description, which gives the estimation method used, the model statement label if specified, the number of cross sections and the number of observations in each cross section, and the order of moving average error process for the DASILVA option
2. the estimates of the underlying error structure parameters
3. the regression parameter estimates and analysis. For each regressor, this includes the name of the regressor, the degrees of freedom, the parameter estimate, the standard error of the estimate, a *t* statistic for testing whether the estimate is significantly different from 0, and the significance probability of the *t* statistic. Whenever possible, the notation of the original reference is followed.

Optionally, PROC TSCSREG prints the following:

4. the covariance and correlation of the resulting regression parameter estimates for each model and assumed error structure
5. the $\hat{\Phi}$ matrix that is the estimated contemporaneous covariance matrix for the PARKS option

ODS Table Names

PROC TSCSREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, "Using the Output Delivery System."

Table 27.1. ODS Tables Produced in PROC TSCSREG

ODS Table Name	**Description**	**Option**
	ODS Tables Created by the MODEL Statement	
ModelDescription	Model Description	
FitStatistics	Fit Statistics	
FixedEffectsTest	F Test for No Fixed Effects	
ParameterEstimates	Parameter Estimates	
CovB	Covariance of Parameter Estimates	
CorrB	Correlations of Parameter Estimates	
VarianceComponents	Variance Component Estimates	
RandomEffectsTest	Hausman Test for Random Effects	
AR1Estimates	First Order Autoregressive Parameter Estimates	
EstimatedPhiMatrix	Estimated Phi Matrix	PARKS
EstimatedAutocovariances	Estimates of Autocovariances	PARKS
	ODS Tables Created by the TEST Statement	
TestResults	Test Results	

Example

Example 27.1. Analyzing Demand for Liquid Assets

In this example, the demand equations for liquid assets are estimated. The demand function for the demand deposits is estimated under three error structures while demand equations for time deposits and savings and loan (S & L) association shares are calculated using the Parks method. The data for seven states (CA, DC, FL, IL, NY, TX, and WA) are selected out of 49 states. Refer to Feige (1964) for data description. All variables were transformed via natural logarithm. The first five observations of the data set A are shown in Output 27.1.1.

```
data a;
   input state $ year d t s y rd rt rs;
   label d = 'Per Capita Demand Deposits'
         t = 'Per Capita Time Deposits'
         s = 'Per Capita S & L Association Shares'
         y = 'Permanent Per Capita Personal Income'
         rd = 'Service Charge on Demand Deposits'
         rt = 'Interest on Time Deposits'
         rs = 'Interest on S & L Association Shares';
datalines;
   ... data lines are omitted ...
;

proc print data=a(obs=5);
run;
```

Output 27.1.1. A Sample of Liquid Assets Data

```
Obs  state  year     d        t        s        y        rd       rt       rs

  1   CA    1949  6.2785   6.1924   4.4998   7.2056   -1.0700  0.1080  1.0664
  2   CA    1950  6.4019   6.2106   4.6821   7.2889   -1.0106  0.1501  1.0767
  3   CA    1951  6.5058   6.2729   4.8598   7.3827   -1.0024  0.4008  1.1291
  4   CA    1952  6.4785   6.2729   5.0039   7.4000   -0.9970  0.4492  1.1227
  5   CA    1953  6.4118   6.2538   5.1761   7.4200   -0.8916  0.4662  1.2110
```

The SORT procedure is used to sort the data into the required time series cross-sectional format. Then PROC TSCSREG analyzes the data.

```
proc sort data=a;
   by state year;
run;

title 'Demand for Liquid Assets';
proc tscsreg data=a;
   model d = y rd rt rs / fuller parks dasilva m=7;
   model t = y rd rt rs / parks;
   model s = y rd rt rs / parks;
   id state year;
run;
```

The income elasticities for liquid assets are greater than 1 except for the demand deposit income elasticity (0.692757) estimated by the Da Silva method. In Output 27.1.2, Output 27.1.3 and Output 27.1.4, the coefficient estimates (-0.29094, -0.43591, and -0.27736) of demand deposits (RD) imply that demand deposits increase significantly as the service charge is reduced. The price elasticities (0.227152 and 0.408066) for time deposits (RT) and S & L association shares (RS) have the expected sign and thus an increase in the interest rate on time deposits or S & L shares will increase the demand for the corresponding liquid asset. Demand deposits and S & L shares appear to be substitutes (Output 27.1.2, Output 27.1.3, Output 27.1.4, and Output 27.1.6). Time deposits are also substitutes for S & L shares in the time deposit demand equation (Output 27.1.5), while these liquid assets are independent of each other in Output 27.1.6 (insignificant coefficient estimate of RT, -0.02705). Demand deposits and time deposits appear to be weak complements in Output 27.1.3 and Output 27.1.4, while the cross elasticities between demand deposits and time deposits are not significant in Output 27.1.2 and Output 27.1.5.

Output 27.1.2. Demand for Demand Deposits – Fuller-Battese Method

```
                            Demand for Liquid Assets

                             The TSCSREG Procedure
                      Fuller and Battese Method Estimation

Dependent Variable: d Per Capita Demand Deposits

                               Model Description

                      Estimation Method             Fuller
                      Number of Cross Sections           7
                      Time Series Length                11


                                Fit Statistics

               SSE               0.0795    DFE                  72
               MSE               0.0011    Root MSE         0.0332
               R-Square          0.6786


                         Variance Component Estimates

               Variance Component for Cross Sections     0.03427
               Variance Component for Time Series        0.00026
               Variance Component for Error              0.00111


                               Hausman Test for
                                Random Effects

                               DF    m Value    Pr > m

                                4       5.51    0.2385


                              Parameter Estimates

                               Standard
 Variable      DF  Estimate       Error  t Value  Pr > |t|  Label

 Intercept      1  -1.23606      0.7252    -1.70    0.0926  Intercept
 y              1  1.064058      0.1040    10.23    <.0001  Permanent Per Capita
                                                            Personal Income
 rd             1  -0.29094      0.0526    -5.53    <.0001  Service Charge on
                                                            Demand Deposits
 rt             1  0.039388      0.0278     1.42    0.1603  Interest on Time
                                                            Deposits
 rs             1  -0.32662      0.1140    -2.86    0.0055  Interest on S & L
                                                            Association Shares
```

Output 27.1.3. Demand for Demand Deposits – Parks Method

```
                            Demand for Liquid Assets

                             The TSCSREG Procedure
                            Parks Method Estimation

Dependent Variable: d Per Capita Demand Deposits

                               Model Description

                      Estimation Method              Parks
                      Number of Cross Sections           7
                      Time Series Length                11


                                Fit Statistics

               SSE             73.3696    DFE                 72
               MSE              1.0190    Root MSE        1.0095
               R-Square         0.9263


                              Parameter Estimates

                                 Standard
 Variable      DF  Estimate      Error  t Value  Pr > |t|  Label

 Intercept      1  -2.66565     0.3139    -8.49    <.0001  Intercept
 y              1  1.222569     0.0423    28.87    <.0001  Permanent Per Capita
                                                           Personal Income
 rd             1  -0.43591     0.0201   -21.71    <.0001  Service Charge on
                                                           Demand Deposits
 rt             1  0.041237     0.0210     1.97    0.0530  Interest on Time
                                                           Deposits
 rs             1  -0.26683     0.0654    -4.08    0.0001  Interest on S & L
                                                           Association Shares
```

Output 27.1.4. Demand for Demand Deposits – Da Silva Method

```
                              Demand for Liquid Assets

                               The TSCSREG Procedure
                             Da Silva Method Estimation

Dependent Variable: d Per Capita Demand Deposits

                                Model Description

                     Estimation Method             DaSilva
                     Number of Cross Sections            7
                     Time Series Length                 11
                     Order of MA Error Process           7


                                  Fit Statistics

               SSE          21609.8923    DFE                  72
               MSE            300.1374    Root MSE        17.3245
               R-Square         0.4995


                           Variance Component Estimates

               Variance Component for Cross Sections     0.03063
               Variance Component for Time Series       0.000148


                                  Estimates of
                                 Autocovariances
                              Lag            Gamma

                              0        0.0008558553
                              1        0.0009081747
                              2        0.0008494797
                              3        0.0007889687
                              4        0.0013281983
                              5        0.0011091685
                              6        0.0009874973
                              7        0.0008462601
```

```
                              Demand for Liquid Assets

                               The TSCSREG Procedure
                             Da Silva Method Estimation

Dependent Variable: d Per Capita Demand Deposits

                               Parameter Estimates

                              Standard
 Variable      DF  Estimate      Error  t Value  Pr > |t|  Label

 Intercept      1  1.281084     0.0824    15.55    <.0001  Intercept
 y              1  0.692757    0.00677   102.40    <.0001  Permanent Per Capita
                                                           Personal Income
 rd             1  -0.27736    0.00274  -101.18    <.0001  Service Charge on
                                                           Demand Deposits
 rt             1  0.009378    0.00171     5.49    <.0001  Interest on Time
                                                           Deposits
 rs             1  -0.09942    0.00601   -16.53    <.0001  Interest on S & L
                                                           Association Shares
```

Output 27.1.5. Demand for Time Deposits – Parks Method

```
                            Demand for Liquid Assets

                             The TSCSREG Procedure
                            Parks Method Estimation

Dependent Variable: t Per Capita Time Deposits

                               Model Description

                      Estimation Method              Parks
                      Number of Cross Sections           7
                      Time Series Length                11


                                 Fit Statistics

               SSE              63.3807    DFE                  72
               MSE               0.8803    Root MSE         0.9382
               R-Square          0.9517


                              Parameter Estimates

                                   Standard
 Variable      DF  Estimate       Error  t Value  Pr > |t|  Label

 Intercept      1  -5.33334      0.5007   -10.65    <.0001  Intercept
 y              1  1.516344      0.0810    18.72    <.0001  Permanent Per Capita
                                                            Personal Income
 rd             1  -0.04791      0.0294    -1.63    0.1082  Service Charge on
                                                            Demand Deposits
 rt             1  0.227152      0.0332     6.85    <.0001  Interest on Time
                                                            Deposits
 rs             1  -0.42569      0.1262    -3.37    0.0012  Interest on S & L
                                                            Association Shares
```

Output 27.1.6. Demand for Savings and Loan Shares – Parks Method

```
                            Demand for Liquid Assets

                             The TSCSREG Procedure
                            Parks Method Estimation

Dependent Variable: s Per Capita S & L Association Shares

                               Model Description

                      Estimation Method              Parks
                      Number of Cross Sections           7
                      Time Series Length                11


                                 Fit Statistics

               SSE               71.9675    DFE                  72
               MSE                0.9995    Root MSE         0.9998
               R-Square           0.9017


                              Parameter Estimates

                             Standard
 Variable      DF  Estimate     Error  t Value  Pr > |t|  Label

 Intercept      1  -8.09632    0.7850   -10.31    <.0001  Intercept
 y              1  1.832988    0.1157    15.84    <.0001  Permanent Per Capita
                                                          Personal Income
 rd             1  0.576723    0.0435    13.26    <.0001  Service Charge on
                                                          Demand Deposits
 rt             1  -0.02705    0.0312    -0.87    0.3891  Interest on Time
                                                          Deposits
 rs             1  0.408066    0.1092     3.74    0.0004  Interest on S & L
                                                          Association Shares
```

Acknowledgments

The TSCSREG procedure was developed by Douglas J. Drummond and A. Ronald Gallant, and contributed to the Version 5 SUGI Supplemental Library in 1979.

Dr. Drummond, now deceased, was with the Center for Survey Statistics, Research Triangle Park, NC. Dr. Drummond programmed the Parks and Fuller-Battese methods. Professor Gallant, who is currently with the University of North Carolina at Chapell Hill, programmed the Da Silva method and generously contributed his time to the support of PROC TSCSREG after Dr. Drummond’s death.

The version of PROC TSCSREG documented here was produced by converting the older SUGI Supplemental Library version of the procedure to Version 6 of SAS software. This conversion work was performed by SAS Institute, which now supports the procedure. Although several features were added during the conversion (such as the OUTEST= option, ID statement, and BY statement), credit for the statistical aspects and general design of the TSCSREG procedure belongs to Dr. Drummond and Professor Gallant.

References

Baltagi, B. H. and Chang, Y. (1994), "Incomplete Panels: A Comparative Study of Alternative Estimators for the Unbalanced One-way Error Component Regression Model," *Journal of Econometrics*, 62(2), 67-89.

Buse, A. (1973), "Goodness of Fit in Generalized Least Squares Estimation," *American Statistician*, 27, 106-108.

Da Silva, J.G.C. (1975), "The Analysis of Cross-Sectional Time Series Data," Ph.D. dissertation, Department of Statistics, North Carolina State University.

SAS Institute Inc. (1979), *SAS Technical Report S-106, TSCSREG: A SAS Procedure for the Analysis of Time-Series Cross-Section Data*, Cary, NC: SAS Institute Inc.

Feige, E.L. (1964), *The Demand for Liquid Assets: A Temporal Cross-Section Analysis*, Englewood Cliffs: Prentice-Hall.

Feige, E.L. and Swamy, P.A.V. (1974), "A Random Coefficient Model of the Demand for Liquid Assets," *Journal of Money, Credit, and Banking*, 6, 241-252.

Fuller, W.A. and Battese, G.E. (1974), "Estimation of Linear Models with Crossed-Error Structure," *Journal of Econometrics*, 2, 67-78.

Greene, W.H. (1990), *Econometric Analysis*, New York: Macmillan Publishing Company.

Hausman, J.A. (1978), "Specification Tests in Econometrics," *Econometrica*, 46, 1251-1271.

Hausman, J.A. and Taylor, W.E. (1982), "A Generalized Specification Test," *Economics Letters*, 8, 239-245.

Hsiao, C. (1986), *Analysis of Panel Data*, Cambridge: Cambridge University Press.

Judge, G.G., Griffiths, W.E., Hill, R.C., Lutkepohl, H., and Lee, T.C. (1985), *The Theory and Practice of Econometrics*, Second Edition, New York: John Wiley & Sons.

Kmenta, J. (1971), *Elements of Econometrics*, New York: MacMillan Publishing Company, Inc.

Maddala, G.S. (1977), *Econometrics*, New York: McGraw-Hill Co.

Parks, R.W. (1967), "Efficient Estimation of a System of Regression Equations when Disturbances Are Both Serially and Contemporaneously Correlated," *Journal of the American Statistical Association*, 62, 500-509.

Searle S.R. (1971), "Topics in Variance Component Estimation," *Biometrics*, 26, 1-76.

Seely, J. (1969), "Estimation in Finite-Dimensional Vector Spaces with Application to the Mixed Linear Model," Ph.D. dissertation, Department of Statistics, Iowa State University.

Seely, J. (1970a), "Linear Spaces and Unbiased Estimation," *Annals of Mathematical Statistics*, 41, 1725-1734.

Seely, J. (1970b), "Linear Spaces and Unbiased Estimation - Application to the Mixed Linear Model," *Annals of Mathematical Statistics*, 41, 1735-1748.

Seely, J. and Soong, S. (1971), "A Note on MINQUE's and Quadratic Estimability," Corvallis, Oregon: Oregon State University.

Seely, J. and Zyskind, G. (1971), "Linear Spaces and Minimum Variance Unbiased Estimation," *Annals of Mathematical Statistics*, 42, 691-703.

Theil, H. (1961), *Economic Forecasts and Policy*, Second Edition, Amsterdam: North-Holland, 435-437.

Wansbeek, T., and Kapteyn, Arie (1989), "Estimation of the Error-Components Model with Incomplete Panels," *Journal of Econometrics*, 41, 341-361.

Wu, D. M. (1973), "Alternative Tests of Independence between Stochastic Regressors and Disturbances," *Econometrica*, 41(4), 733-750.

Zellner, A. (1962), "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias," *Journal of the American Statistical Association*, 57, 348-368.

Chapter 28

The TIMESERIES Procedure

Chapter Contents

Chapter 28 The TIMESERIES Procedure

Overview

The TIMESERIES procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format. The procedure can perform trend and seasonal analysis on the transactions. Once the transactional data are accumulated, time domain and frequency domain analysis can be performed on the accumulated time series.

For seasonal analysis of the transaction data, various statistics can be computed for each season. For trend analysis of the transaction data, various statistics can be computed for each time period. The analysis is similar to applying the MEANS procedure of Base SAS software to each season or time period of concern.

Once the transactional data are accumulated to form a time series and any missing values are interpreted, the accumulated time series can be functionally transformed using log, square root, logistic, or Box-Cox transformations. The time series can be further transformed using simple and/or seasonal differencing. After functional and difference transformations have been applied, the accumulated and transformed time series can be stored in an output data set. This working time series can then be analyzed further using various time series analysis techniques provided by this procedure or other SAS/ETS procedures.

Time series analyses performed by the TIMESERIES procedure include:

- Descriptive (Global) Statistics
- Seasonal Decomposition/Adjustment Analysis
- Correlation Analysis
- Cross-correlation Analysis

All results of the transactional or time series analysis can be stored in output data sets or printed using the Output Delivery System (ODS).

Experimental graphics are now available with the TIMESERIES procedure. For more information, see the "ODS Graphics" section on page 1603.

The TIMESERIES procedure can process large amounts of time-stamped transactional data. Therefore, the analysis results are useful for large-scale time series analysis or (temporal) data mining. All of the results can be stored in output data sets in either a time series format (default) or in coordinate format (transposed). The time series format is useful for preparing the data for subsequent analysis using other SAS/ETS procedures. For example, the working time series can be further analyzed, modeled, and forecast using other SAS/ETS procedures. The coordinate format is useful when using this procedure with SAS/STAT procedures or Enterprise Miner. For example, clustering time-stamped transactional data can be achieved by using the results of this procedure with the clustering procedures of SAS/STAT and the nodes of Enterprise Miner.

The EXPAND procedure can be used for the frequency conversion and transformations of time series output from this procedure.

Getting Started

This section outlines the use of the TIMESERIES procedure and gives a cursory description of some of the analysis techniques that can be performed on time-stamped transactional data.

Given an input data set that contains numerous transaction variables recorded over time at no specific frequency, the TIMESERIES procedure can form time series as follows:

```
PROC TIMESERIES DATA=<input-data-set> OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   VAR <time-series-variables>;
RUN;
```

The TIMESERIES procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats using the Output Delivery System (ODS). The following examples are more fully illustrated in the "Examples" section on page 1606.

Time-stamped transactional data are often recorded at no fixed interval. Analysts often want to use time series analysis techniques that require fixed-time intervals. Therefore, the transactional data must be accumulated to form a fixed-interval time series.

Suppose that a bank wishes to analyze the transactions associated with each of its customers over time. Further, suppose that the data set WORK.TRANSACTIONS contains four variables related to these transactions: CUSTOMER, DATE, WITHDRAWAL, and DEPOSITS. The following examples illustrate possible ways to analyze these transactions using the TIMESERIES procedure.

The following statements illustrate how to use the TIMESERIES procedure to accumulate time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (WITHDRAWALS and DEPOSITS).

```
proc timeseries data=transactions out=timeseries;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The OUT=TIMESERIES option specifies that the resulting time series data for each customer is to be stored in the data set WORK.TIMESERIES. The INTERVAL=DAY option specifies that the transactions are to accumulated on a daily basis. The ACCUMULATE=TOTAL option specifies that the sum of the transactions are to be accumulated. Once the transactional data are accumulated into a time series format, many of the procedures provided with SAS/ETS software can be used to analyze the time series data.

For example, the ARIMA procedure can be used to model and forecast each customer's transactions using an ARIMA(0,1,1)(0,1,1)$_s$ model (where the number of seasons is s=7 days in a week) using the following statements:

```
proc arima data=timeseries;
   identify var=withdrawals(1,7) noprint;
   estimate q=(1,7) outest=estimates noprint;
   forecast id=date interval=day out=forecasts;
quit;
```

The OUTEST=ESTIMATES data set will contain the parameter estimates of the model specified. The OUT=FORECASTS data set will contain forecasts based on the model specified. See the ARIMA procedure for more detail.

A single set of transactions can be very large and must be summarized in order to analyze them effectively. Analysts often want to examine transactional data for trends and seasonal variation. To analyze transactional data for trends and seasonality, statistics must be computed for each time period and season of concern. For each observation, the time period and season must be determined and the data must be analyzed based on this determination.

The following statements illustrate how to use the TIMESERIES procedure to perform trend and seasonal analysis of time-stamped transactional data.

```
proc timeseries data=transactions out=out
     outseason=season outtrend=trend;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

Since the INTERVAL=DAY option is specified, the length of the seasonal cycle is seven (7) where the first season is Sunday and the last season is Saturday. The output data set specified by the OUTSEASON=SEASON option contains the seasonal statistics for each day of the week by each customer. The output data set specified by the OUTTREND=TREND option contains the trend statistics for each day of the calendar by each customer.

Often it is desired to seasonally decompose into seasonal, trend, cycle, and irregular components or seasonally adjust a time series. These techniques describe how the changing seasons influence the time series.

The following statements illustrate how to use the TIMESERIES procedure to perform seasonal adjustment/decomposition analysis of time-stamped transactional data.

```
proc timeseries data=transactions out=out outdecomp=decompose;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The output data set specified by the OUTDECOMP=DECOMPOSE data set contains the decomposed/adjusted time series for each customer.

A single time series can be very large. Often, a time series must be summarized with respect to time lags in order to be efficiently analyzed. Analysts often want to analyze time series data using time domain techniques. These techniques help describe how a current observation is related to the past observations with respect to the time (season) lag.

The following statements illustrate how to use the TIMESERIES procedure to perform time domain analysis of time-stamped transactional data.

```
proc timeseries data=transactions out=out outcorr=timedomain;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The output data set specified by the OUTCORR=TIMEDOMAIN data set contains the time domain statistics by each customer.

Syntax

The following statements are used with the TIMESERIES procedure.

PROC TIMESERIES *options*;
  BY *variables*;
  CORR *statistics-list / options*;
  CROSSCORR *statistics-list / options*;
  DECOMP *component-list / options*;
  SEASON *statistics-list / options*;
  TREND *statistics-list / options*;
  VAR *variable-list / options*;
  CROSSVAR *variable-list / options*;
  ID *variable* **INTERVAL=** *interval options*;

Functional Summary

The statements and options controlling the TIMESERIES procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify variables to analyze	VAR	
specify cross-variables to analyze	CROSSVAR	
specify the time ID variable	ID	
specify correlation options	CORR	
specify cross-correlation optons	CROSSCORR	
specify decomposition optons	DECOMP	
specify seasonal statistics optons	SEASON	
specify trend statistics optons	TREND	
Data Set Options		
specify the input data set	PROC TIMESERIES	DATA=
specify the output data set	PROC TIMESERIES	OUT=
specify correlations output data set	PROC TIMESERIES	OUTCORR=
specify cross-correlations output data set	PROC TIMESERIES	OUTCROSSCORR=
specify decomposition output data set	PROC TIMESERIES	OUTDECOMP=
specify seasonal statistics output data set	PROC TIMESERIES	OUTSEASON=
specify summary statistics output data set	PROC TIMESERIES	OUTSUM=
specify trend statistics output data set	PROC TIMESERIES	OUTTREND=
Accumulation and Seasonality Options		
specify accumulation frequency	ID	INTERVAL=
specify length of seasonal cycle	PROC TIMESERIES	SEASONALITY=
specify interval alignment	ID	ALIGN=
specify time ID variable values are not sorted	ID	NOTSORTED
specify starting time ID value	ID	START=
specify ending time ID value	ID	END=
specify accumulation statistic	ID, VAR, CROSSVAR	ACCUMULATE=

Description	Statement	Option
specify missing value interpretation	ID, VAR, CROSSVAR	SETMISS=
Time-Stamped Data Seasonal Statistics Options		
specify the form of the output data set	SEASON	TRANSPOSE=
Time-Stamped Data Trend Statistics Options		
specify the form of the output data set	TREND	TRANSPOSE=
specify the number of time periods to be stored	TREND	NPERIODS=
Time Series Transformation Options		
specify simple differencing	VAR, CROSSVAR	DIF=
specify seasonal differencing	VAR, CROSSVAR	SDIF=
specify transformation	VAR, CROSSVAR	TRANSFORM=
Time Series Correlation Options		
specify the list of lags	CORR	LAGS=
specify the number of lags	CORR	NLAG=
specify the number of parameters	CORR	NPARMS=
specify the form of the output data set	CORR	TRANSPOSE=
Time Series Cross-correlation Options		
specify the list of lags	CROSSCORR	LAGS=
specify the number of lags	CROSSCORR	NLAG=
specify the form of the output data set	CROSSCORR	TRANSPOSE=
Time Series Decomposition Options		
specify mode of decomposition	DECOMP	MODE=
specify the Hodrick-Prescott filter parameter	DECOMP	LAMBDA=
specify the number of time periods to be stored	DECOMP	NPERIODS=
specify the form of the output data set	DECOMP	TRANSPOSE=
Printing Control Options		
specify time ID format	ID	FORMAT=
specify printed output	PROC TIMESERIES	PRINT=
specify detailed printed output	PROC TIMESERIES	PRINTDETAILS
Miscellaneous Options		
specify that analysis variables are processed in sorted order	PROC TIMESERIES	SORTNAMES

Description	Statement	Option
limits error and warning messages	PROC TIMESERIES	MAXERROR=

PROC TIMESERIES Statement

PROC TIMESERIES *options***;**

The following options can be used in the PROC TIMESERIES statement.

DATA= *SAS-data-set*
names the SAS data set containing the input data for the procedure to create time series. If the DATA= option is not specified, the most recently created SAS data set is used.

MAXERROR= *number*
limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERRORS=50. This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

OUT= *SAS-data-set*
names the output data set to contain the the time series variables specified in the subsequent VAR statements. If an ID variable is specified, it will also be included in the OUT= data set. The values are accumulated based on the ID statement INTERVAL= and/ or ACCUMULATE= option. The OUT= data set is particularly useful when you wish to further analyze, model, or forecast the resulting time series with other SAS/ETS procedures.

OUTCORR= *SAS-data-set*
names the output data set to contain the univariate time domain statistics.

OUTCROSSCORR= *SAS-data-set*
names the output data set to contain the cross-correlation statistics.

OUTDECOMP= *SAS-data-set*
names the output data set to contain the decomposed and/or seasonally adjusted time series.

OUTSEASON= *SAS-data-set*
names the output data set to contain the seasonal statistics. The statistics are computed for each season as specified by the INTERVAL= option or the SEASONALITY= option. The OUTSEASON= data set is particularly useful when analyzing transactional data for seasonal variations.

OUTSUM= *SAS-data-set*
names the output data set to contain the descriptive statistics. The descriptive statistics are based on the accumulated time series when the ACCUMULATE= or

SETMISSING= options are specified. The OUTSUM= data set is particularly useful when analyzing large numbers of series and a summary of the results are needed.

OUTTREND= *SAS-data-set*
names the output data set to contain the trend statistics. The statistics are computed for each time period as specified by the INTERVAL= option. The OUTTREND= data set is particularly useful when analyzing transactional data for trends.

PRINT= *option* **| (***options***)**
specifies the printed output desired. By default, the TIMESERIES procedure produces no printed output. The following printing options are available:

DECOMP	prints the seasonal decomposition/adjustment table. (OUTDECOMP= data set)
SEASONS	prints the seasonal statistics table. (OUTSEASON= data set)
DESCSTATS	prints the descriptive statistics for the accumulated time series. (OUTSUM= data set)
SUMMARY	prints the descriptive statistics table for all time series. (OUTSUM= data set)
TRENDS	prints the trend statistics table. (OUTTREND= data set)

For example, PRINT=SEASONS prints the seasonal statistics. The PRINT= option produces printed output for these results utilizing the Output Delivery System (ODS). The PRINT= option produces results similar to the data sets listed next to the above options in parenthesis.

PRINTDETAILS
specifies that output requested with the PRINT= option be printed in greater detail.

SEASONALITY= *number*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three time periods forms a seasonal cycle. By default, the length of the seasonal cycle is one (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is twelve.

SORTNAMES
specifies that the variables specified in the VAR statements are processed in sorted order by the variable names.

BY Statement

BY *variables;*

A BY statement can be used with PROC TIMESERIES to obtain separate analyses for groups of observations defined by the BY variables.

CORR Statement

CORR *statistics / options*;

A CORR statement can be used with the TIMESERIES procedure to specify options related to time domain analysis of the accumulated time series. Only one CORR statement is allowed.

The following time domain statistics are available:

LAG	Time lag
N	Number of Variance Products
ACOV	Autocovariances
ACF	Autocorrelations
ACFSTD	Autocorrelation Standard Errors
ACF2STD	Indicates ACF Beyond Two Standard Errors
ACFNORM	Normalized Autocorrelations
ACFPROB	Autocorrelation Probabilities
ACFLPROB	Autocorrelation Log Probabilities
PACF	Partial Autocorrelations
PACFSTD	Partial Autocorrelation Standard Errors
PACF2STD	Indicates PACF Beyond Two Standard Errors
PACFNORM	Partial Normalized Autocorrelations
PACFPROB	Partial Autocorrelation Probabilities
PACFLPROB	Partial Autocorrelation Log Probabilities
IACF	Inverse Autocorrelations
IACFSTD	Inverse Autocorrelation Standard Errors
IACF2STD	Indicates IACF Beyond Two Standard Errors
IACFNORM	Normalized Inverse Autocorrelations
IACFPROB	Inverse Autocorrelation Probabilities
IACFLPROB	Inverse Autocorrelation Log Probabilities
WN	White Noise Test Statistics
WNPROB	White Noise Test Probabilities
WNLPROB	White Noise Test Log Probabilities

If none of the correlation statistics are specified, the default is as follows:

```
corr lag n acov acf acfstd pacf pacfstd iacf iacfstd wn wnprob;
```

The following options can be specified in the CORR statement following the slash (/):

NLAG= *number*
LAGS= *(numlist)*
specifies the number of lags or list of lags to be stored in OUTCORR= data set or printed. The default is 24 or three times the length of the seasonal cycle whichever is smaller.

NPARMS= *number*
specifies the number of parameters used in the model that created the residual time series. The number of parameters determines the degrees of freedom associated with the Ljung-Box statistics. The default is NPARMS=0.

TRANSPOSE=NO | YES
TRANSPOSE=YES specifies that the OUTCORR= data set is recorded with the lags as the column names instead of the correlation statistics as the column names. The TRANSPOSE=NO option is particularly useful for graphing the correlation results using SAS/GRAPH procedures. The TRANSPOSE=YES option is particularly useful for analyzing the correlation results using other SAS/STAT procedures or Enterprise Miner. The default is TRANSPOSE=NO.

CROSSCORR Statement

CROSSCORR *statistics / options*;

A CROSSCORR statement can be used with the TIMESERIES procedure to specify options related to cross-correlation analysis of the accumulated time series. Only one CROSSCORR statement is allowed.

The following time domain statistics are available:

LAG	Time lag
N	Number of Variance Products
CCOV	Cross-Covariances
CCF	Cross-correlations
CCFSTD	Cross-correlation Standard Errors
CCF2STD	Indicates CCF Beyond Two Standard Errors
CCFNORM	Normalized Cross-correlations
CCFPROB	Cross-correlations Probabilities
CCFLPROB	Cross-correlations Log Probabilities

If none of the correlation statistics are specified, the default is as follows:

```
crosscorr lag n ccov ccf ccfstd;
```

The following options can be specified in the CROSSCORR statement following the slash (/):

NLAG= *number*
LAGS= *(numlist)*

specifies the number of lags or list of lags to be stored in OUTCROSSCORR= data set or printed. The default is 24 or three times the length of the seasonal cycle whichever is smaller.

TRANSPOSE=NO | YES

TRANSPOSE=YES specifies that the OUTCROSSCORR= data set is recorded with the lags as the column names instead of the cross-correlation statistics as the column names. The TRANSPOSE=NO option is particularly useful for graphing the cross-correlation results using SAS/GRAPH procedures. The TRANSPOSE=YES option is particularly useful for analyzing the cross-correlation results using other SAS/STAT procedures or Enterprise Miner. The default is TRANSPOSE=NO.

DECOMP Statement

DECOMP *components / options*;

A DECOMP statement can be used with the TIMESERIES procedure to specify options related to classical seasonal decomposition of the time series data. Only one DECOMP statement is allowed. The options specified affects all variables specified in the VAR statements. Decomposition can be performed only when the length of the seasonal cycle implied by the INTERVAL= option or specified by the SEASONALITY= option is greater than one.

The following seasonal decomposition components are available:

ORIG|ORIGINAL	Original Series
TCC|TRENDCYCLE	Trend-Cycle Component
SIC|SEASONIRREGULAR	Seasonal-Irregular Component
SC|SEASONAL	Seasonal Component
SCSTD	Seasonal Component Standard Errors
TCS|TRENDCYCLESEASON	Trend-Cycle-Seasonal Component
IC|IRREGULAR	Irregular Component
SA|ADJUSTED	Seasonal Adjusted
PCSA	Percent Change Seasonal Adjusted
TC	Trend Component
CC|CYCLE	Cycle Component

If none of the components are specified, the default is as follows:

```
decomp orig tcc sc ic sa;
```

The following options can be specified in the DECOMP statement following the slash (/):

MODE=ADD|ADDITIVE
MODE=MULT|MULTIPLICATIVE
MODE=LOGADD|LOGADDITIVE
MODE=PSEUDOADD|PSEUDOADDITIVE
MODE=MULTORADD

specifies the type of decomposition is to be used to decompose the time series. Multiplicative and log additive decomposition requires a positive-valued time series. If the accumulated time series contains nonpositive values and the MODE=MULT or MODE=LOGADD option is specified, an error results. Pseudo-additive decomposition requires a nonnegative-valued time series. If the accumulated time series contains negative values and the MODE=PSEUDOADD option is specified, an error results. The MODE=MULTORADD option specifies that multiplicative decomposition is used when the accumulated time series contains only positive values, that pseudo-additive decomposition is used when the accumulated time series contains only nonnegative values, and that additive decomposition is used otherwise. The default is MODE=MULTORADD.

LAMBDA= *number*

specifies the Hodrick-Prescott filter parameter for trend-cycle decomposition. The default is LAMBDA=1600. If filtering is not specified this option is ignored.

NPERIODS= *number*

specifies the number of time periods to be stored in the OUTDECOMP= data set when the TRANSPOSE=YES option is specified. If the TRANSPOSE=NO option is specified, the NPERIODS= option is ignored. If the NPERIODS= option is positive the first or beginning time periods are recorded. If the NPERIODS= option is negative the last or ending time periods are recorded. The NPERIODS= option specifies the number of OUTDECOMP= data set variables to contain the seasonal decomposition and is therefore limited to the maximum allowable number of SAS variables. If the number of time periods exceeds this limit a warning is printed in the log and the number periods stored is reduced to the limit.

If NPERIODS= option is not specified, all of the periods specified between the ID statement START= and END= options are stored. If either of the START= or END= options are not specified, the default magnitude is the seasonality specified by the TIMESERIES statement SEASONALITY= option or implied by the INTERVAL= option. If only the START= option is specified, the default sign is positive. If only the END= option is specified, the default sign is negative.

```
/* NPERIODS=10 because there are ten months between
   the specified start and end dates */
   id date interval=month accumulate=total
      start='01JAN2000'D end='01OCT2000'D;
   decomp / transpose=yes;

/* NPERIODS=10 because there are ten months between
   the specified start and end dates */
```

```
      id date interval=month accumulate=total
         start='01JAN2000'D end='01OCT2000'D;
      decomp / transpose=yes nperiods=100;

   /* NPERIODS=12 because there are twelve months in a year
      and only the start date is specified */
      id date interval=month accumulate=total
         start='01JAN2000'D;
      decomp / transpose=yes;

   /* NPERIODS=-12 because there are twelve months in a year
      and only the end date is specified */
      id date interval=month accumulate=total
         end='01OCT2000'D;
      decomp / transpose=yes;
```

TRANSPOSE=NO | YES

TRANSPOSE=YES specifies that the OUTDECOMP= data set is recorded with the time periods as the column names instead of the statistics as the column names. The first and last time period stored in the OUTDECOMP= data set corresponds to the period of the ID statement START= option and END= option, respectively. If only the ID statement END= option is specified, the last time ID value of each accumulated time series corresponds to the last time period column. If only the ID statement START= option is specified, the first time ID value of each accumulated time series corresponds to the first time period column. If neither the START= option or END= option is specified with the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. The TRANSPOSE=NO option is particularly useful for analyzing the decomposition results using other SAS/ETS procedures or graphing the decomposition results using SAS/GRAPH procedures such as the GPLOT procedure. The TRANSPOSE=YES option is particularly useful for analyzing the decomposition results using other SAS/STAT procedures or Enterprise Miner. The default is TRANSPOSE=NO.

ID Statement

ID *variable* **INTERVAL=** *interval options*;

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The information specified affects all variables specified in subsequent VAR statements. If the ID statement is specified, the INTERVAL= option must also be specified. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

The following options can be used with the ID statement.

ACCUMULATE= *option*

specifies how the data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent analysis.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations coinciding with a particular time period (e.g., time-stamped transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following options determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

NONE	No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option.
TOTAL	Observations are accumulated based on the total sum of their values.
AVERAGE | AVG	Observations are accumulated based on the average of their values.
MINIMUM | MIN	Observations are accumulated based on the minimum of their values.
MEDIAN | MED	Observations are accumulated based on the median of their values.
MAXIMUM | MAX	Observations are accumulated based on the maximum of their values.
N	Observations are accumulated based on the number of non-missing observations.
NMISS	Observations are accumulated based on the number of missing observations.
NOBS	Observations are accumulated based on the number of observations.
FIRST	Observations are accumulated based on the first of their values.
LAST	Observations are accumulated based on the last of their values.
STDDEV |STD	Observations are accumulated based on the standard deviation of their values.
CSS	Observations are accumulated based on the corrected sum of squares of their values.
USS	Observations are accumulated based on the uncorrected sum of squares of their values.

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The DETAILS section describes accumulation in greater detail.

ALIGN= *option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

END= *option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This START= and END= option can be used to ensure that data associated within each BY group contains the same number of observations.

FORMAT= *format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied from the INTERVAL= option.

INTERVAL= *interval*

specifies the frequency of the accumulated time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations.

NOTSORTED

specifies that the time ID values are not in sorted order. The TIMESERIES procedure will sort the data with respect to the time ID prior to analysis.

SETMISSING= *option* **|** *number*

specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series. If a number is specified, missing values are set to number. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, a SETMISSING=0 should be used. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. The following options can also be used to determine how missing values are assigned:

MISSING	Missing values are set to missing. This is the default option.
AVERAGE | AVG	Missing values are set to the accumulated average value.
MINIMUM | MIN	Missing values are set to the accumulated minimum value.
MEDIAN | MED	Missing values are set to the accumulated median value.

MAXIMUM | MAX	Missing values are set to the accumulated maximum value.
FIRST	Missing values are set to the accumulated first non-missing value.
LAST	Missing values are set to the accumulated last non-missing value.
PREVIOUS | PREV	Missing values are set to the previous period's accumulated non-missing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next period's accumulated non-missing value. Missing values at the end of the accumulated series remain missing.

START= *option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, the series is prepended with missing values. If the first time ID variable value is less than the START= value, the series is truncated. This START= and END= option can be used to ensure that data associated with each by group contains the same number of observations.

SEASON Statement

SEASON *statistics / options*;

A SEASON statement can be used with the TIMESERIES procedure to specify options related to seasonal analysis of the time-stamped transactional data. Only one SEASON statement is allowed. The options specified affects all variables specified in the VAR statements. Seasonal analysis can be performed only when the length of the seasonal cycle implied by the INTERVAL= option or specified by the SEASONALITY= option is greater than one.

The following seasonal statistics are available:

NOBS	Number of Observations
N	Number of Non-Missing Observations
NMISS	Number of Missing Observations
MINIMUM	Minimum Value
MAXIMUM	Maximum Value
RANGE	Range Value
SUM	Summation Value
MEAN	Mean Value
STDDEV	Standard Deviation
CSS	Corrected Sum of Squares
USS	Uncorrected Sum of Squares

MEDIAN	Median Value

If none of the season statistics is specified, the default is as follows:

```
season n min max mean std;
```

The following options can be specified in the SEASON statement following the slash (/):

TRANSPOSE=NO | YES

TRANSPOSE=YES specifies that the OUTSEASON= data set is recorded with the seasonal indices as the column names instead of the statistics as the column names. The TRANSPOSE=NO option is particularly useful for graphing the seasonal analysis results using SAS/GRAPH procedures. The TRANSPOSE=YES option is particularly useful for analyzing the seasonal analysis results using other SAS/STAT procedures or Enterprise Miner. The default is TRANSPOSE=NO.

TREND Statement

TREND *statistics / options* **;**

A TREND statement can be used with the TIMESERIES procedure to specify options related to trend analysis of the time-stamped transactional data. Only one TREND statement is allowed. The options specified affects all variables specified in the VAR statements.

The following trend statistics are available:

NOBS	Number of Observations
N	Number of Non-Missing Observations
NMISS	Number of Missing Observations
MINIMUM	Minimum Value
MAXIMUM	Maximum Value
RANGE	Range Value
SUM	Summation Value
MEAN	Mean Value
STDDEV	Standard Deviation
CSS	Corrected Sum of Squares
USS	Uncorrected Sum of Squares
MEDIAN	Median Value

If none of the trend statistics is specified, the default is as follows:

```
trend n min max mean std;
```

The following options can be specified in the TREND statement following the slash (/):

NPERIODS= *number*

Specifies the number of time periods to be stored in the OUTTREND= data set when the TRANSPOSE option is specified. If the TRANSPOSE option is not specified, the NPERIODS= option is ignored. The NPERIODS= option specifies the number of OUTTREND= data set variables to contain the trend statistics and is therefore limited to the maximum allowable number of SAS variables.

If NPERIODS= option is not specified, all of the periods specified between the ID statement START= and END= options are stored. If either of the START= or END= options are not specified, the default is the seasonality specified by the ID statement SEASONALITY= option or implied by the INTERVAL= option. If the seasonality is zero, the default is NPERIODS=5.

TRANSPOSE=NO | YES

TRANSPOSE=YES specifies that the OUTTREND= data set is recorded with the time periods as the column names instead of the statistics as the column names. The first and last time period stored in the OUTTREND= data set corresponds to the period of the ID statement START= option and END= option, respectively. If only the ID statement END= option is specified, the last time ID value of each accumulated time series corresponds to the last time period column. If only the ID statement START= option is specified, the first time ID value of each accumulated time series corresponds to the first time period column. If neither the START= option or END= option is specified with the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. The TRANSPOSE=NO option is particularly useful for analyzing the trend analysis results using other SAS/ETS procedures or graphing the trend analysis results using SAS/GRAPH procedures such as the GPLOT procedure. The TRANSPOSE=YES option is particularly useful for analyzing the trend analysis results using other SAS/STAT procedures or Enterprise Miner. The default is TRANSPOSE=YES.

VAR and CROSSVAR Statements

VAR *variable-list / options*;

CROSSVAR *variable-list / options*;

The VAR and CROSSVAR statement lists the numeric variables in the DATA= data set whose values are to be accumulated to form the time series.

An input data set variable can be specified in only one VAR or CROSSVAR statement. Any number of VAR and CROSSVAR statements can be used. The following options can be used with the VAR and CROSSVAR statement.

ACCUMULATE= *option*

specifies how the data set observations are accumulated within each time period for the variables listed in the VAR statement. If the ACCUMULATE= option is not specified in the VAR statement, accumulation is determined by the ACCUMULATE=

option of the ID statement. See the ID statement ACCUMULATE= option for more details.

DIF= *numlist*
specifies the differencing to be applied to the accumulated time series.

SDIF= *numlist*
specifies the seasonal differencing to be applied to the accumulated time series.

SETMISS= *option* **|** *number*
SETMISSING= *option* **|** *number*
option specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the VAR statement. If the SETMISSING= option is not specified in the VAR statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRANSFORM= *option*
specifies the time series transformation to be applied to the accumulated time series. The following transformations are provided:

NONE	No transformation is applied. This option is the default.
LOG	Logarithmic transformation
SQRT	Square-root transformation
LOGISTIC	Logistic transformation
BOXCOX(*n*)	Box-Cox transformation with parameter number where number is between -5 and 5

When the TRANSFORM= option is specified the time series must be strictly positive.

Details

The TIMESERIES procedure can be used to perform trend and seasonal analysis on transactional data. For trend analysis, various sample statistics are computed for each time period defined by the time ID variable and INTERVAL= option. For seasonal analysis, various sample statistics are computed for each season defined by the INTERVAL= or the SEASONALITY= option. For example, if the transactional data ranges from June 1990 to January 2000 and the data are to be accumulated on a monthly basis, then the trend statistics are computed for every month: June 1990, July 1990, ..., January 2000. The seasonal statistics are computed for each season: January, February, ..., December.

The TIMESERIES procedure can be used to form time series data from transactional data. The accumulated time series can then be analyzed using time series techniques.

1. Accumulation	ACCUMULATE= option
2. Missing Value Interpretation	SETMISSING= option
3. Time Series Transformation	TRANSFORM= option
4. Time Series Differencing	DIF= and SDIF= option
5. Descriptive Statistics	OUTSUM= option, PRINT=DESCSTATS
6. Seasonal Decomposition	DECOMP statement, OUTDECOMP= option
7. Correlation Analysis	CORR statement, OUTCORR= option
8. Cross-correlation Analysis	CROSSCORR statement, OUTCROSSCORR= option

Accumulation

If the ACCUMULATE= option is specified, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the above observations fall within three time periods of March 1999, April 1999, and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified:

```
O1MAR1999    40
O1APR1999     .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified:

```
O1MAR1999    20
O1APR1999     .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified:

```
O1MAR1999    10
O1APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified:

```
O1MAR1999    20
01APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified:

```
O1MAR1999    30
O1APR1999     .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified:

```
O1MAR1999    10
O1APR1999     .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified:

```
O1MAR1999    30
O1APR1999     .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified:

```
01MAR1999    14.14
01APR1999    .
01MAY1999    17.32
```

As can be seen from the above examples, even though the data set observations contained no missing values, the accumulated time series may have missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no value, i.e. zero. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

Time Series Transformation

There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log — is the logarithmic transformation.

$$w_t = \ln(\mathrm{y_t})$$

Logistic — is the logistic transformation.

$$w_t = \ln(\mathrm{cy_t}/(1 - \mathrm{cy_t}))$$

where the scaling factor c is

$$c = (1 - 10^{-6})10^{-\mathrm{ceil}(\log_{10}(\max(\mathrm{y_t})))}$$

and $\mathrm{ceil(x)}$ is the smallest integer greater than or equal to *x*.

Square Root — is the square root transformation.

$$w_t = \sqrt{y_t}$$

Box Cox — is the Box-Cox transformation.

$$w_t = \begin{cases} \frac{y_t^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(\mathrm{y_t}), & \lambda = 0 \end{cases}$$

More complex time series transformations can be performed using the EXPAND procedure of SAS/ETS.

Time Series Differencing

After optionally transforming the series, the accumulated series can be simply or seasonally differenced using the VAR statement DIF= and SDIF= option. For example, suppose y_t is a monthly time series, the following examples of the DIF= and SDIF= options demonstrate how to simply and seasonally difference the time series.

```
dif=(1) sdif=(1)
dif=(1,12)
```

Additionally assuming y_t is strictly positive, the VAR statement TRANSFORM= option, DIF= and SDIF= options can be combined.

Descriptive Statistics

Descriptive statistics can be computed from the working series by specifying the OUTSUM= option or the PRINT=DESCSTATS.

Seasonal Decomposition

Seasonal decomposition/analysis can be performed on the working series by specifying the OUTDECOMP= option, the PRINT=DECOMP option, or one of the PLOT= option associated with decomposition. The DECOMP statement enables you to specify options related to decomposition. The TIMESERIES procedure uses classical decomposition. More complex seasonal decomposition/adjustment analysis can be performed using the X11 or the X12 procedure of SAS/ETS.

The DECOMP statement MODE= option determines the mode of the seasonal adjustment decomposition to be performed. There are four modes: multiplicative (MODE=MULT), additive (MODE=ADD), pseudo-additive (MODE=PSEUDOADD), and log-additive (MODE=LOGADD) decomposition. The default is MODE=MULTORADD which specifies MODE=MULT for series that are strictly positive, MODE=PSEUDOADD for series that are nonnegative, and MODE=ADD for series that are not.

When MODE=LOGADD is specified, the components are exponentiated to the original metric.

The DECOMP statement LAMBDA= option specifies the Hodrick-Prescott filter parameter. The default is LAMBDA=1600. The Hodrick-Prescott filter is used to decompose the trend-cycle component into the trend component and cycle component in an additive fashion. A smaller parameter assigns less significance to the cycle, that is, LAMBDA=0 implies no cycle component.

The notation and keywords associated with seasonal decomposition/adjustment analysis are defined as follows:

Table 28.1. Seasonal Adjustment Formulas

Component	**Keyword**	**MULT**	**ADD**	**LOGADD**	**PSEUDOADD**
Original	ORIGINAL	$O_t = TC_tS_tI_t$	$O_t = TC_t + S_t + I_t$	$log(O_t) = TC_t + S_t + I_t$	$O_t = TC_t(S_t + I_t - 1)$
Trend-Cycle Component	TCC	Centered Moving Average of O_t	Centered Moving Average of O_t	Centered Moving Average of $log(O_t)$	Centered Moving Average of O_t
Seasonal-Irregular Component	SIC	$SI_t = S_tI_t = O_t/TC_t$	$SI_t = S_t + I_t = O_t - TC_t$	$SI_t = S_t + I_t = log(O_t) - TC_t$	$SI_t = S_t + I_t - 1 = O_t/TC_t$
Seasonal Component	SC	Seasonal Averages of SI_t	Seasonal Averages of SI_t	Seasonal Averages of SI_t	Seasonal Averages of SI_t
Irregular Component	IC	$I_t = SI_t/S_t$	$I_t = SI_t - S_t$	$I_t = SI_t - S_t$	$I_t = SI_t - S_t + 1$
Trend-Cycle-Seasonal Component	TCS	$TCS_t = TC_tS_t = O_t/I_t$	$TCS_t = TC_t + S_t = O_t - I_t$	$TCS_t = TC_t + S_t = O_t - I_t$	$TCS_t = TC_tS_t$
Trend Component	TC	$T_t = TC_t - C_t$	$T_t = TC_t - C_t$	$T_t = TC_t - C_t$	$T_t = TC_t - C_t$
Cycle Component	CC	$C_t = TC_t - T_t$	$C_t = TC_t - T_t$	$C_t = TC_t - T_t$	$C_t = TC_t - T_t$
Seasonal Adjusted	SA	$SA_t = O_t/S_t = TC_tI_t$	$SA_t = O_t - S_t = TC_t + I_t$	$SA_t = O_t/exp(S_t) = exp(TC_t + I_t)$	$SA_t = TC_tI_t$

Correlation Analysis

Correlation analysis can be performed on the working series by specifying the OUTCORR= option or one of the PLOT= options associated with correlation. The CORR statement enables you to specify options related to correlation analysis.

Autocovariance Statistics

LAGS	$h \in \{0, \ldots, H\}$
N	N_h is the number of observed products at lag h ignoring missing values
ACOV	$\hat{\gamma}(h) = \frac{1}{T}\sum_{t=h+1}^{T}(y_t - \overline{y})(y_{t-h} - \overline{y})$
ACOV	$\hat{\gamma}(h) = \frac{1}{N_h}\sum_{t=h+1}^{T}(y_t - \overline{y})(y_{t-h} - \overline{y})$ embedded missing values

Autocorrelation Statistics

ACF	$\hat{\rho}(h) = \hat{\gamma}(h)/\hat{\gamma}(0)$
ACFSTD	$Std(\hat{\rho}(h)) = \sqrt{\frac{1}{T}\left(1 + 2\sum_{j=1}^{h-1}\hat{\rho}(j)^2\right)}$
ACFNORM	$Norm(\hat{\rho}(h)) = \hat{\rho}(h)/Std(\hat{\rho}(h))$
ACFPROB	$Prob(\hat{\rho}(h)) = 2\,(1 - \Phi\,(\lvert Norm(\hat{\rho}(h))\rvert))$
ACFLPROB	$LogProb(\hat{\rho}(h)) = -\log_{10}(Prob(\hat{\rho}(h))$
ACF2STD	$Flag(\hat{\rho}(h)) = \begin{cases} 1 & \hat{\rho}(h) > 2Std(\hat{\rho}(h)) \\ 0 & -2Std(\hat{\rho}(h)) < \hat{\rho}(h) < 2Std(\hat{\rho}(h)) \\ -1 & \hat{\rho}(h) < -2Std(\hat{\rho}(h)) \end{cases}$

Partial Autocorrelation Statistics

PACF	$\hat{\varphi}(h) = \Gamma_{(0,h-1)}\{\gamma_j\}_{j=1}^{h}$
PACFSTD	$Std(\hat{\varphi}(h)) = 1/\sqrt{N_0}$
PCFNORM	$Norm(\hat{\varphi}(h)) = \hat{\varphi}(h)/Std(\hat{\varphi}(h))$
PACFPROB	$Prob(\hat{\varphi}(h)) = 2\,(1 - \Phi\,(\lvert Norm(\hat{\varphi}(h))\rvert))$
PACFLPROB	$LogProb(\hat{\varphi}(h)) = -\log_{10}(Prob(\hat{\varphi}(h))$
PACF2STD	$Flag(\hat{\varphi}(h)) = \begin{cases} 1 & \hat{\varphi}(h) > 2Std(\hat{\varphi}(h)) \\ 0 & -2Std(\hat{\varphi}(h)) < \hat{\varphi}(h) < 2Std(\hat{\varphi}(h)) \\ -1 & \hat{\varphi}(h) < -2Std(\hat{\varphi}(h)) \end{cases}$

Inverse Autocorrelation Statistics

IACF	$\hat{\theta}(h)$
IACFSTD	$Std(\hat{\theta}(h)) = 1/\sqrt{N_0}$
IACFNORM	$Norm(\hat{\theta}(h)) = \hat{\theta}(h)/Std(\hat{\theta}(h))$
IACFPROB	$Prob(\hat{\theta}(h)) = 2\left(1 - \Phi\left(\lvert Norm(\hat{\theta}(h))\rvert\right)\right)$

IACFLPROB	$LogProb(\hat{\theta}(h)) = -\log_{10}(Prob(\hat{\theta}(h))$
IACF2STD	$Flag(\hat{\theta}(h)) = \begin{cases} 1 & \hat{\theta}(h) > 2Std(\hat{\theta}(h)) \\ 0 & -2Std(\hat{\theta}(h)) < \hat{\theta}(h) < 2Std(\hat{\theta}(h)) \\ -1 & \hat{\theta}(h) < -2Std(\hat{\theta}(h)) \end{cases}$

White Noise Statistics

WN	$Q(h) = T(T+2)\sum_{j=1}^{h} \rho(j)^2/(T-j)$
WN	$Q(h) = \sum_{j=1}^{h} N_j \rho(j)^2$ embedded missing values
WNPROB	$Prob(Q(h)) = \chi_{\max(1,h-p)}(Q(h))$
WNLPROB	$LogProb(Q(h)) = -\log_{10}(Prob(Q(h))$

Cross-correlation Analysis

Cross-correlation analysis can be performed on the working series by specifying the OUTCROSSCORR= option or one of the CROSSPLOT= options associated with cross-correlation. The CROSSCORR statement enables you to specify options related to cross-correlation analysis.

Cross-correlation Statistics

LAGS	$h \in \{0, \ldots, H\}$
N	N_h is the number of observed products at lag h ignoring missing values
CCOV	$\hat{\gamma}_{x,y}(h) = \frac{1}{T}\sum_{t=h+1}^{T}(x_t - \overline{x})(y_{t-h} - \overline{y})$
CCOV	$\hat{\gamma}_{x,y}(h) = \frac{1}{N_h}\sum_{t=h+1}^{T}(x_t - \overline{x})(y_{t-h} - \overline{y})$ embedded missing values
CCF	$\hat{\rho}_{x,y}(h) = \hat{\gamma}_{x,y}(h)/\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}$
CCFSTD	$Std(\hat{\rho}_{x,y}(h)) = 1/\sqrt{N_0}$
CCFNORM	$Norm(\hat{\rho}_{x,y}(h)) = \hat{\rho}_{x,y}(h)/Std(\hat{\rho}_{x,y}(h))$
CCFPROB	$Prob(\hat{\rho}_{x,y}(h)) = 2\left(1 - \Phi\left(\lvert Norm(\hat{\rho}_{x,y}(h))\rvert\right)\right)$
CCFLPROB	$LogProb(\hat{\rho}_{x,y}(h)) = -\log_{10}(Prob(\hat{\rho}_{x,y}(h))$
CCF2STD	$Flag(\hat{\rho}_{x,y}(h)) = \begin{cases} 1 & \hat{\rho}_{x,y}(h) > 2Std(\hat{\rho}_{x,y}(h)) \\ 0 & -2Std(\hat{\rho}_{x,y}(h)) < \hat{\rho}_{x,y}(h) < 2Std(\hat{\rho}_{x,y}(h)) \\ -1 & \hat{\rho}_{x,y}(h) < -2Std(\hat{\rho}_{x,y}(h)) \end{cases}$

Data Set Output

The TIMESERIES procedure can create the OUT=, and OUTSUM= data sets. In general, these data sets will contain the variables listed in the BY statement. In general, if an analysis step related to an output data step fails, the values of this step are not recorded or are set to missing in the related output data set, and appropriate error and/or warning messages are recorded in the log.

OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, and VAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR statements are accumulated based on the ACCUMULATE= option and missing values are interpreted based on the SETMISSING= option.

OUTCORR= Data Set

The OUTCORR= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTCORR= data set records the correlations for each variable specified in a VAR statement.

When the CORR statement TRANSPOSE=NO option is specified, the variable *names* are related to correlation statistics specified in the CORR statement options; the variable *values* are related to the NLAG= or LAGS= options.

NAME	Variable name
LAG	Time lag
N	Number of Variance Products
ACOV	Autocovariances
ACF	Autocorrelations
ACFSTD	Autocorrelation Standard Errors
ACF2STD	Indicates ACF Beyond Two Standard Errors
ACFNORM	Normalized Autocorrelations
ACFPROB	Autocorrelation Probabilities
ACFLPROB	Autocorrelation Log Probabilities
PACF	Partial Autocorrelations
PACFSTD	Partial Autocorrelation Standard Errors
PACF2STD	Indicates PACF Beyond Two Standard Errors
PACFNORM	Partial Normalized Autocorrelations
PACFPROB	Partial Autocorrelation Probabilities
PACFLPROB	Partial Autocorrelation Log Probabilities
IACF	Inverse Autocorrelations
IACFSTD	Inverse Autocorrelation Standard Errors
IACF2STD	Indicates IACF Beyond Two Standard Errors
IACFNORM	Normalized Inverse Autocorrelations
IACFPROB	Inverse Autocorrelation Probabilities
IACFLPROB	Inverse Autocorrelation Log Probabilities
WN	White Noise Test Statistics

WNPROB	White Noise Test Probabilities
WNLPROB	White Noise Test Log Probabilities

The above correlation statistics are computed for each specified time lag.

When the CORR statement TRANSPOSE=YES option is specified, the variable *values* are related to correlation statistics specified in the CORR statement; the variable *names* are related to the NLAG= or LAGS= options.

NAME	Variable name
STAT	Correlation statistic name
LABEL	Correlation statistic label
LAG*h*	Correlation statistics for lag *h*

OUTCROSSCORR= Data Set

The OUTCROSSCORR= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTCROSSCORR= data set records the cross-correlations for each variable specified in a VAR statement.

When the CROSSCORR statement TRANSPOSE=NO option is specified, the variable *names* are related to cross-correlation statistics specified in the CROSSCORR statement options; the variable *values* are related to the NLAG= or LAGS= options.

NAME	Variable name
CROSS	Cross Variable name
LAG	Time lag
N	Number of Variance Products
CCOV	Cross-Covariances
CCF	Cross-correlations
CCFSTD	Cross-correlation Standard Errors
CCF2STD	Indicates CCF Beyond Two Standard Errors
CCFNORM	Normalized Cross-correlations
CCFPROB	Cross-correlation Probabilities
CCFLPROB	Cross-correlation Log Probabilities

The above cross-correlation statistics are computed for each specified time lag.

When the CROSSCORR statement TRANSPOSE=YES option is specified, the variable *values* are related to cross-correlation statistics specified in the CROSSCORR statement; the variable *names* are related to the NLAG= or LAGS= options.

NAME	Variable name
CROSS	Cross Variable name
STAT	Cross-correlation statistic name
LABEL	Cross-correlation statistic label
LAG*h*	Cross-correlation statistics for lag *h*

OUTDECOMP= Data Set

The OUTDECOMP= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTDECOMP= data set records the seasonal decomposition/adjustments for each variable specified in a VAR statement.

When the DECOMP statement TRANSPOSE=NO option is specified, the variable *names* are related to decomposition/adjustments specified in the DECOMP statement; the variable *values* are related to the INTERVAL= and SEASONALITY= options.

NAME	Variable name
MODE	Mode of decomposition
TIMEID	Time ID values
SEASON	Seasonal index
ORIGINAL	Original series values
TCC	Trend-Cycle Component
SIC	Seasonal-Irregular Component
SC	Seasonal Component
SCSTD	Seasonal Component Standard Errors
TCS	Trend-Cycle-Seasonal Component
IC	Irregular Component
SA	Seasonal Adjusted Component
PCSA	Percent Change Seasonal Adjusted Component
TC	Trend Component
CC	CYCLE Component

The above decomposition components are computed for each time period.

When the DECOMP statement TRANSPOSE=YES option is specified, the variable *values* are related to decomposition/adjustments specified in the DECOMP statement; the variable *names* are related to the INTERVAL=, SEASONALITY=, and NPERIODS= options.

NAME	Variable name
MODE	Mode of decomposition name

COMP	Decomposition component name
LABEL	Decomposition component label
PERIOD*t*	Decomposition component value for time period *t*

OUTSEASON= Data Set

The OUTSEASON= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTSEASON= data set records the seasonal statistics for each variable specified in a VAR statement.

When the SEASON statement TRANSPOSE=NO option is specified, the variable *names* are related to seasonal statistics specified in the SEASON statement; the variable *values* are related to the INTERVAL= or SEASONALITY= options.

NAME	Variable name
TIMEID	Time ID values
SEASON	Seasonal index
NOBS	Number of Observations
N	Number of Non-Missing Observations
NMISS	Number of Missing Observations
MINIMUM	Minimum Value
MAXIMUM	Maximum Value
RANGE	Maximum Value
SUM	Summation Value
MEAN	Mean Value
STDDEV	Standard Deviation
CSS	Corrected Sum of Squares
USS	Uncorrected Sum of Squares
MEDIAN	Median Value

The above statistics are computed for each season.

When the SEASON statement TRANSPOSE=YES option is specified, the variable *values* are related to seasonal statistics specified in the SEASON statement; the variable *names* are related to the INTERVAL= or SEASONALITY= options.

NAME	Variable name
STAT	Season statistic name
LABEL	Season statistic name
SEASON*s*	Season statistic value for season *s*

OUTSUM= Data Set

The OUTSUM= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTSUM= data set records the descriptive statistics for each variable specified in a VAR statement.

Variables related to descripive statistics are based on the ACCUMULATE= and SETMISSING= options:

NAME	Variable name
STATUS	Status flag that indicates whether the requested analyses were successful
NOBS	Number of Observations
N	Number of Non-Missing Observations
NMISS	Number of Missing Observations
MINIMUM	Minimum Value
MAXIMUM	Maximum Value
AVG	Average Value
STDDEV	Standard Deviation

The OUTSUM= data set contains the descriptive statistics of the (accumulated) time series.

OUTTREND= Data Set

The OUTTREND= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTTREND= data set records the trend statistics for each variable specified in a VAR statement.

When the TREND statement TRANSPOSE=NO option is specified, the variable *names* are related to trend statistics specified in the TREND statement; the variable *values* are related to the INTERVAL= or SEASONALITY= options.

NAME	Variable name
TIMEID	Time ID values
SEASON	Seasonal index
NOBS	Number of Observations
N	Number of Non-Missing Observations
NMISS	Number of Missing Observations
MINIMUM	Minimum Value
MAXIMUM	Maximum Value
RANGE	Maximum Value
SUM	Summation Value

MEAN	Mean Value
STDDEV	Standard Deviation
CSS	Corrected Sum of Squares
USS	Uncorrected Sum of Squares
MEDIAN	Median Value

The above statistics are computed for each time period.

When the TREND statement TRANSPOSE=YES option is specified, the variable *values* related to trend statistics specified in the TREND statement; the variable *name* are related to the INTERVAL=, SEASONALITY=, and NPERIODS= options.

NAME	Variable name
STAT	Trend statistic name
LABEL	Trend statistic name
PERIOD*t*	Trend statistic value for time period *t*

Printed Output

The TIMESERIES procedure optionally produces printed output for these results utilizing the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options associated with the PROC TIMESERIES statement. In general, if an analysis step related to printed output fails, the values of this step are not printed and appropriate error and/or warning messages are recorded in the log. The printed output is similar to the output data set and these similarities are described below.

PRINT=DECOMP

prints the seasonal decomposition similar to the OUTDECOMP= data set.

PRINT=DESCSTATS

prints a table of descriptive statistics for each variable.

PRINT=SEASONS

prints the seasonal statistics similar to the OUTSEASON= data set.

PRINT=SUMMARY

prints the summary statistics similar to the OUTSUM= data set.

PRINT=TRENDS

prints the trend statistics similar to the OUTTREND= data set.

PRINTDETAILS

The PRINTDETAILS option prints each table with greater detail.

Specifically, if PRINT=SEASONS and the PRINTDETAILS options are specified, all seasonal statistics are printed.

ODS Table Names

The table below relates the PRINT= options to ODS tables:

Table 28.2. ODS Tables Produced in PROC TIMESERIES

ODS Table Name	Description	Option
ODS Tables Created by the PRINT=DECOMP option		
SeasonalDecomposition	Seasonal Decomposition	
ODS Tables Created by the PRINT=DESCSTATS option		
DescStats	Descriptive Statistics	
ODS Tables Created by the PRINT=SEASONS option		
GlobalStatistics	Global Statistics	
SeasonStatistics	Season Statistics	
ODS Tables Created by the PRINT=SUMMARY option		
StatisticsSummary	Statistics Summary	
ODS Tables Created by the PRINT=TRENDS option		
GlobalStatistics	Global Statistics	
TrendStatistics	Trend Statistics	

The tables are related to a single series within a BY group.

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the TIMESERIES procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the PLOT= or CROSSPLOT= option in the TIMESERIES statement according to the following syntax. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

PLOT= *option* **| (***options***)**

specifies the univariate graphical output desired. By default, the TIMESERIES procedure produces no graphical output. The following plotting options are available:

SERIES	plots time series graphics. (OUT= data set)
RESIDUAL	plots residual time series graphics. (OUT= data set)
CORR	plots correlation panel graphics. (OUTCORR= data set)

ACF	plots autocorrelation function graphics. (OUTCORR= data set)
PACF	plots partial autocorrelation function graphics. (OUTCORR= data set)
IACF	plots inverse autocorrelation function graphics. (OUTCORR= data set)
WN	plots white noise graphics. (OUTCORR= data set)
DECOMP	plots seasonal adjustment panel graphics. (OUTDECOMP= data set)
TCS	plots the trend-cycle-seasonal component graphics. (OUTDECOMP= data set)
TCC	plots the trend-cycle component graphics. (OUTDECOMP= data set)
SIC	plots the seasonal-irregular component graphics. (OUTDECOMP= data set)
SC	plots the seasonal component graphics. (OUTDECOMP= data set)
SA	plots the seasonal adjusted graphics. (OUTDECOMP= data set)
PCSA	plots the percent change seasonal adjusted graphics. (OUTDECOMP= data set)
IC	plots the irregular component graphics. (OUTDECOMP= data set)
TC	plots the trend component graphics. (OUTDECOMP= data set)
CC	plots the cycle component graphics. (OUTDECOMP= data set)
ALL	Same as PLOT=(SERIES ACF PACF IACF WN).

For example, PLOT=SERIES plots the time series. The PLOT= option produces graphical output for these results utilizing the Output Delivery System (ODS). The PLOT= option produces results similar to the data sets listed next to the above options in parenthesis.

CROSSPLOT= *option* **|** **(***options***)**

specifies the cross-variable graphical output desired. By default, the TIMESERIES procedure produces no graphical output. The following plotting options are available:

SERIES	plots time series graphics. (OUT= data set)
CCF	plots autocorrelation function graphics. (OUTCORR= data set)
ALL	Same as PLOT=(SERIES CCF).

For example, CROSSPLOT=SERIES plots the two time series. The CROSSPLOT= option produces graphical output for these results utilizing the Output Delivery System (ODS). The CROSSPLOT= option produces results similar to the data sets listed next to the above options in parenthesis.

ODS Graph Names

PROC TIMESERIES assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 28.3.

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the PLOT= or CROSSPLOT= option in the TIMESERIES statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 28.3. ODS Graphics Produced by PROC TIMESERIES

ODS Graph Name	Plot Description	Statement	PLOT= Option
ACFPlot	Autocorrelation Function	TIMESERIES	PLOT=ACF
ACFNORMPlot	Standardized Autocorrelation Function	TIMESERIES	PLOT=ACF
CCFNORMPlot	Standardized Cross-correlation Function	TIMESERIES	CROSSPLOT=CCF
CCFPlot	Cross-correlation Function	TIMESERIES	CROSSPLOT=CCF
CorrelationPlots	Correlation Graphics Panel	TIMESERIES	PLOT=CORR
CrossSeriesPlot	Cross Series Plot	TIMESERIES	CROSSPLOT=SERIES
CycleComponentPlot	Cycle Component	TIMESERIES	PLOT=CC
DecompositionPlots	Decomposition Graphiics Panel	TIMESERIES	PLOT=DECOMP
IACFPlot	Inverse Autocorrelation Function	TIMESERIES	PLOT=IACF
IACFNORMPlot	Standardized Inver-se Autocorrelation Function	TIMESERIES	PLOT=IACF
IrregularComponentPlot	Irregular Component	TIMESERIES	PLOT=IC
PACFPlot	Partial Autocorrelation Function	TIMESERIES	PLOT=PACF
PACFNORMPlot	Standardized Partial Autocorrelation Function	TIMESERIES	PLOT=PACF
PercentChangeAdjustedplot	Percent-Change Seasonally Adjusted	TIMESERIES	PLOT=SA
ResidualPlot	Residual Time Series Plot	TIMESERIES	PLOT=RESIDUAL
SeasonalAdjusted	Seasonally Adjusted	TIMESERIES	PLOT=SA
SeasonalComponentPlot	Seasonal Component	TIMESERIES	PLOT=SC
SeasonalIrregularComponentPlot	Seasonal-Irregular Component	TIMESERIES	PLOT=SIC
SeriesPlot	Time Series Plot	TIMESERIES	PLOT=SERIES

Table 28.3. (continued)

ODS Graph Name	Plot Description	Statement	Option
TrendComponentPlot	Trend Component	TIMESERIES	PLOT=TC
TrendCycleComponentPlot	Trend-Cycle Component	TIMESERIES	PLOT=TCC
TrendCycleSeasonalPlot	Trend-Cycle-Seasonal Component	TIMESERIES	PLOT=TCS
WhiteNoiseLogProb	White Noise Log Probability	TIMESERIES	PLOT=WN
WhiteNoiseProbability	White Noise Probability	TIMESERIES	PLOT=WN

Examples

Example 28.1. Accumulating Transactional Data into Time Series Data

This example illustrates the accumulation of time-stamped transactional data that has been recorded at no particular frequency into time series data at a specific frequency using the TIMESERIES procedure. Once the time series is created, the various SAS/ETS procedures related to time series analysis, seasonal adjustment/decomposition, modeling, and forecasting can be used to further analyze the time series data.

Suppose that the input data set WORK.RETAIL contains variables STORE and TIMESTAMP and numerous other numeric transaction variables. The BY variable STORE contains values that break up the transactions into groups (BY groups). The time ID variable TIMESTAMP contains SAS date values recorded at no particular frequency. The other data set variables contain the numeric transaction values to be analyzed. It is further assumed that the input data set is sorted by the variables STORE and TIMESTAMP.

The following statements form monthly time series from the transactional data based on the median value (ACCUMULATE=MEDIAN) of the transactions recorded with each time period. Also, the accumulated time series values for time periods with no transactions are set to zero instead of missing (SETMISS=0) and only transactions recorded between the first day of 1998 (START=’01JAN1998’D) and last day of 2000 (END=’31JAN2000’D).are considered and if needed extended to include this range.

```
proc timeseries data=work.retail out=mseries;
   by store;
   id timestamp interval=month accumulate=median setmiss=0
      start=’01jan1998’d
      end  =’31dec2000’d;
   var _ALL_;
run;
```

The monthly time series data are stored in the data WORK.MSERIES. Each BY group associated with the BY variable STORE will contain an observation for each of the 36 months associated with the years 1998, 1999, and 2000. Each observation will contain the variable STORE, TIMESTAMP, and each of the analysis variables in the input data set.

Once each set of transactions has been accumulated to form corresponding time series, accumulated time series can be analyzed using various time series analysis techniques. For example, exponentially weighted moving averages can be used to smooth each series. The following statements use the EXPAND procedure to smooth the analysis variable named STOREITEM.

```
proc expand data=mseries out=smoothed from=month;
   by store;
   id date;
   convert storeitem=smooth / transform=(ewma 0,1);
run;
```

The decomposed series are stored in the data set WORK.SMOOTHED. The variable SMOOTH contains the smoothed series.

If the time ID variable TIMESTAMP contained SAS datetime values instead of SAS date values, the INTERVAL= , START=, and END= option must be changed accordingly and the following statements could be used.

```
proc timeseries data=work.retail out=tseries;
   by store;
   id timestamp interval=dtmonth accumulate=median setmiss=0
      start='01jan1998:00:00:00'dt
      end  ='31dec2000:00:00:00'dt;
   var _ALL_;
run;
```

The monthly time series data are stored in the data WORK.TSERIES and the time ID values use a SAS datetime representation.

Example 28.2. Trend and Seasonal Analysis

This example illustrates trend and seasonal analysis of time-stamped transactional data using the TIMESERIES procedure.

Suppose that the data set SASHELP.AIR contains two variable DATE and AIR. The variable DATE contains sorted SAS date values recorded at no particular frequency. The variable AIR contains the transaction values to be analyzed.

The following statements accumulate the transactional data on an average basis to form a quarterly time series and perform trend and seasonal analysis on the transactions.

```
proc timeseries data=sashelp.air out=series
                outtrend=trend outseason=season print=seasons;
   id date interval=qtr accumulate=avg;
   var air;
run;
```

The time series is stored in the data set WORK.SERIES, the trend statistics are stored in the data set WORK.TREND, and the seasonal statistics are stored in the data set WORK.SEASON. Additionally, the seasonal statistics are printed (PRINT=SEASONS) and the results of the seasonal analysis are shown in Output 28.2.1.

Output 28.2.1. Seasonal Statistics Table

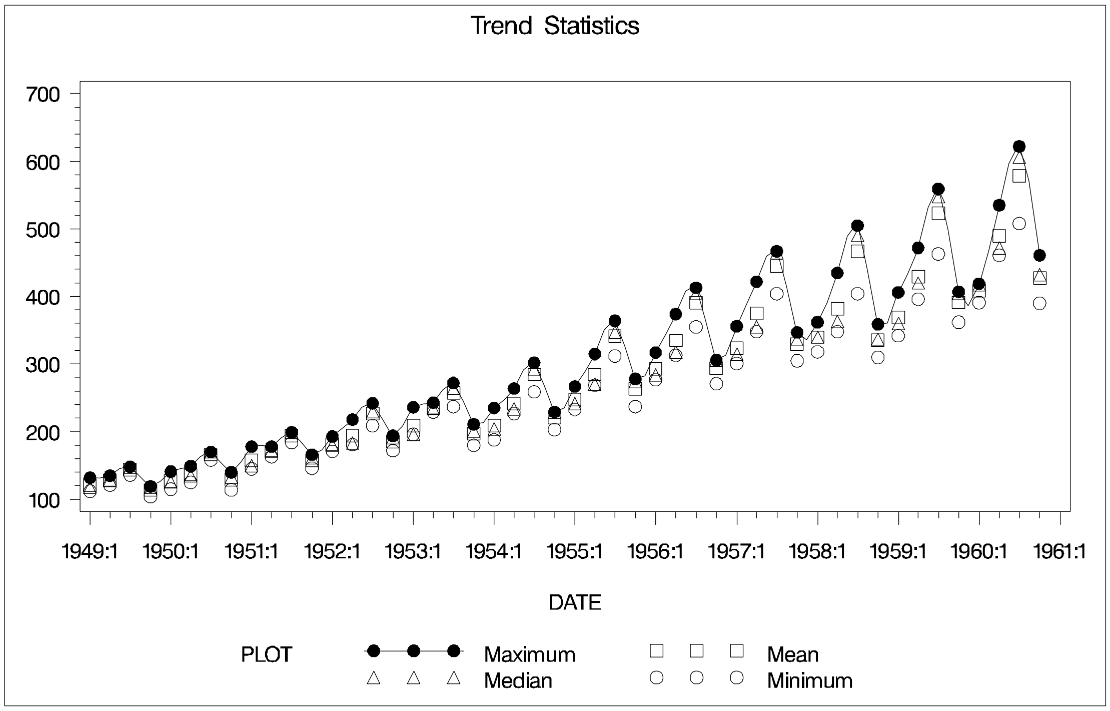
```
                              The TIMESERIES Procedure

                         Season Statistics for Variable AIR

Season                                                                    Standard
 Index        N      Minimum      Maximum          Sum         Mean      Deviation

     1       36     112.0000     419.0000      8963.00     248.9722       95.65189
     2       36     121.0000     535.0000     10207.00     283.5278      117.61839
     3       36     136.0000     622.0000     12058.00     334.9444      143.97935
     4       36     104.0000     461.0000      9135.00     253.7500      101.34732
```

Using the trend statistics stored in the WORK.TREND data set, the following statements plot various trend statistics associated with each time period over time.

```
title1 "Trend Statistics";
legend1 value=("Maximum" "Mean" "Median" "Minimum");
symbol interpol=spline;
axis2 label=none;
proc gplot data=trend;
   plot max    *date
        mean   *date
        median *date
        min    *date
        / overlay vaxis=axis2 legend=legend1;
 run;
```

The results of this trend analysis are shown in Output 28.2.2.

Output 28.2.2. Trend Statistics Plot

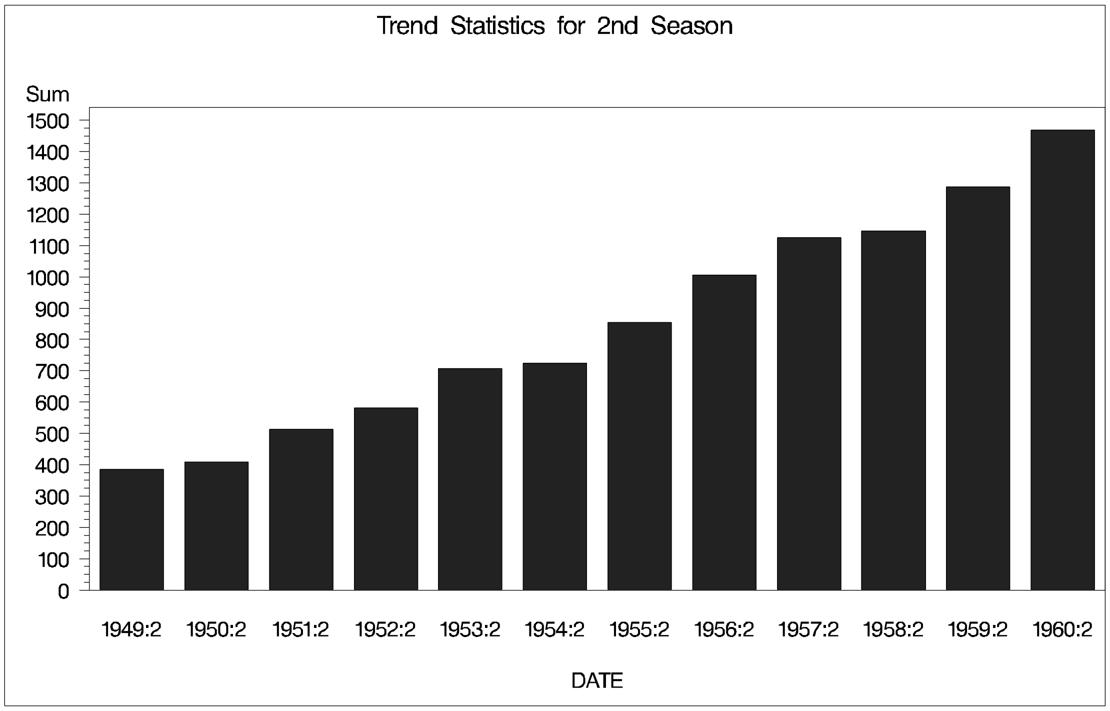


Using the trend statistics stored in the WORK.TREND data set, the following statements chart the sum of the transactions associated with each time period for the second season over time.

```
title1 "Trend Statistics for 2nd Season";
proc gchart data=trend;
   where _season_ = 2;
   vbar date / sumvar=sum discrete;
run;
quit;
```

The results of this trend analysis are shown in Output 28.2.3.

Output 28.2.3. Trend Statistics Plot

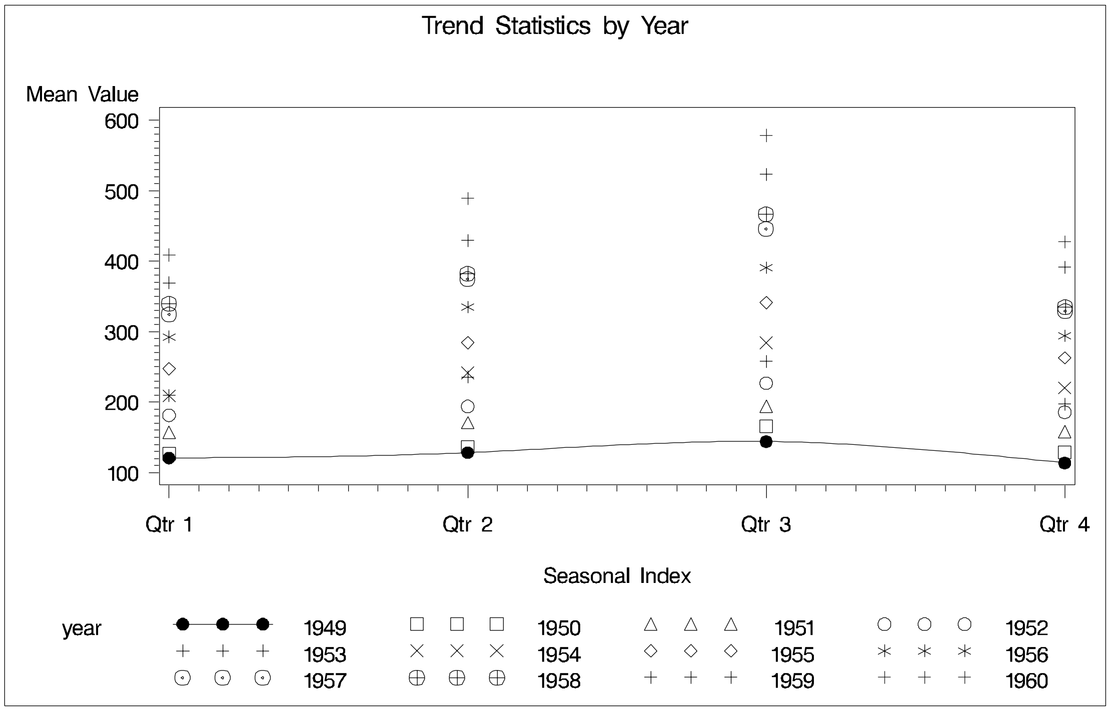


Using the trend statistics stored in the WORK.TREND data set, the following statements plot the mean of the transactions associated with each time period by each year over time.

```
data trend;
   set trend;
   year = year(date);
run;

title1 "Trend Statistics by Year";
symbol interpol=spline;
axis1 value=('Qtr 1' 'Qtr 2' 'Qtr 3' 'Qtr 4' );
proc gplot data=trend;
  plot mean*_season_=year / haxis=axis1;
run;
```

The results of this trend analysis are shown in Output 28.2.4.

Output 28.2.4. Trend Statistics

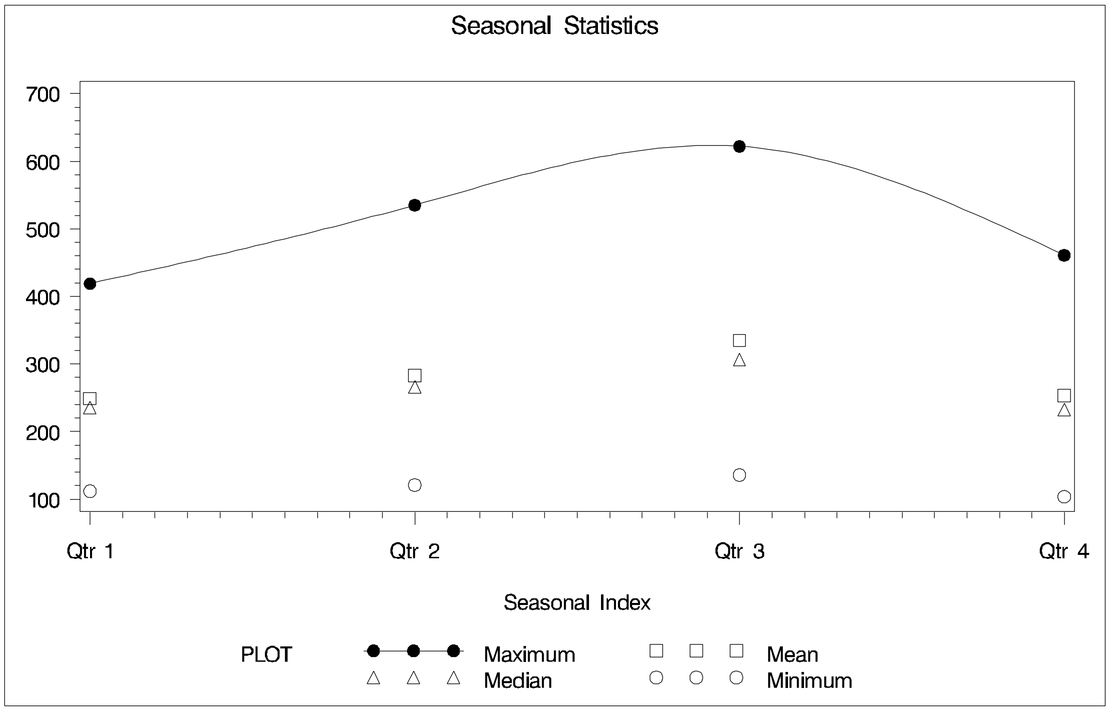


Using the season statistics stored in the WORK.SEASON data set, the following statements plot various season statistics for each season.

```
title1 "Seasonal Statistics";
legend1 value=("Maximum" "Mean" "Median" "Minimum");
symbol interpol=spline;
axis1 value=('Qtr 1' 'Qtr 2' 'Qtr 3' 'Qtr 4' );
axis2 label=none;
proc gplot data=season;
   plot max    *_season_
        mean   *_season_
        median *_season_
        min    *_season_
        / overlay haxis=axis1 vaxis=axis2 legend=legend1;
run;
```

The results of this seasonal analysis are shown in Output 28.2.5.

Output 28.2.5. Seasonal Statistics Plot



Example 28.3. Illustration of ODS Graphics (Experimental)

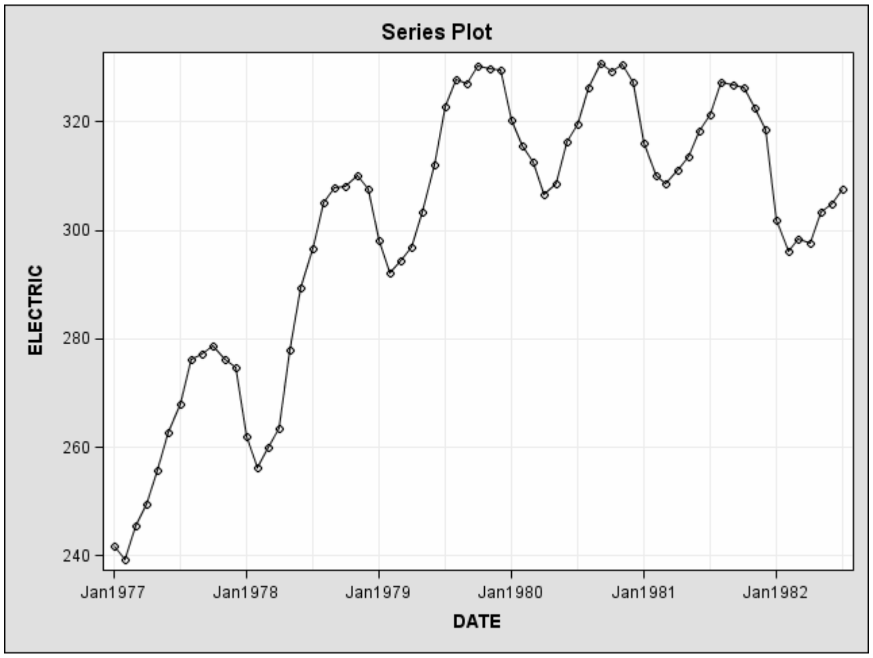
This example illustrates the use of experimental ODS graphics.

The following statements utilize the SASHELP.WORKERS data set to study the time series of electrical workers, and its interaction with the series of masonry workers. The series plot, the correlation panel, the seasonal adjustment panel, and all cross-series plots are requested. Output 28.3.1 through Output 28.3.4 show a selection of the plots created.

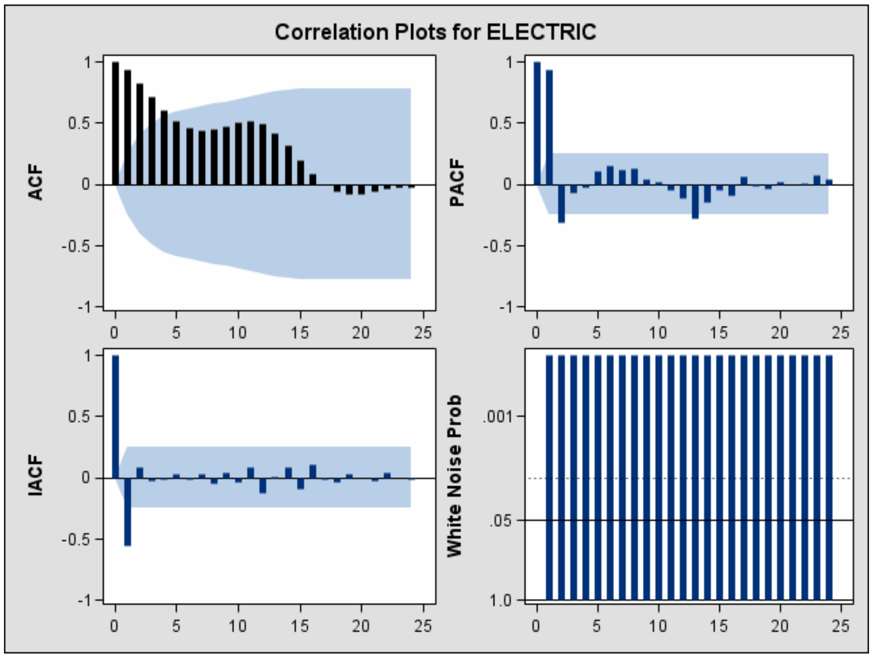
The graphical displays are requested by specifying the experimental ODS GRAPHICS statement and the experimental PLOT= or CROSSPLOT= options in the PROC TIMESERIES statement. For general information about ODS graphics, see Chapter 9, “Statistical Graphics Using ODS.” For specific information about the graphics available in the TIMESERIES procedure, see the “ODS Graphics” section on page 1603.

```
ods html;
ods graphics on;

title "Illustration of ODS Graphics";
proc timeseries data=sashelp.workers out=_null_
   plot=(series corr decomp)
   crossplot=all;
   id date interval=month;
   var electric;
   crossvar masonry;
run;

ods graphics off;
ods html close;
```

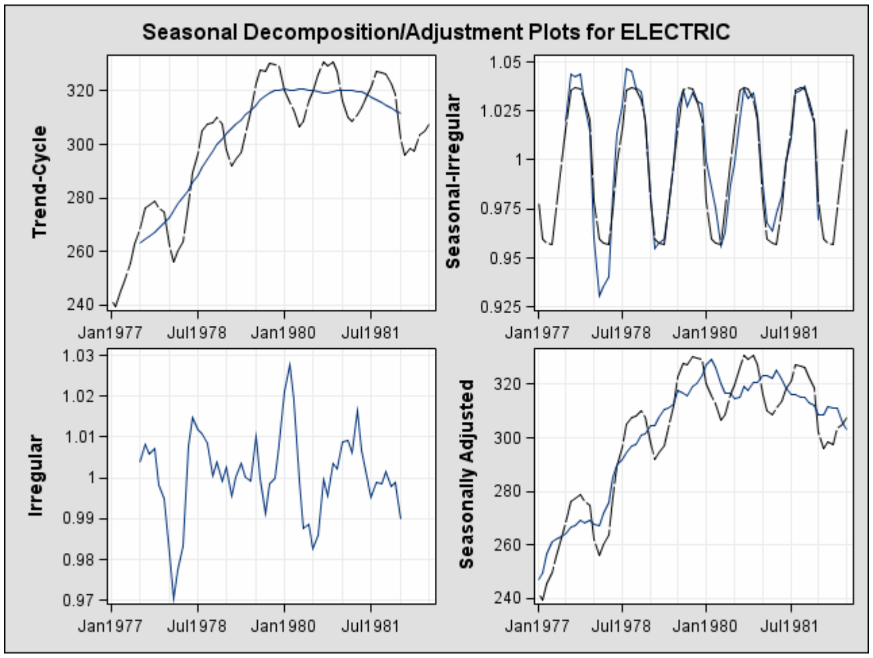
Output 28.3.1. Series Plot (Experimental)



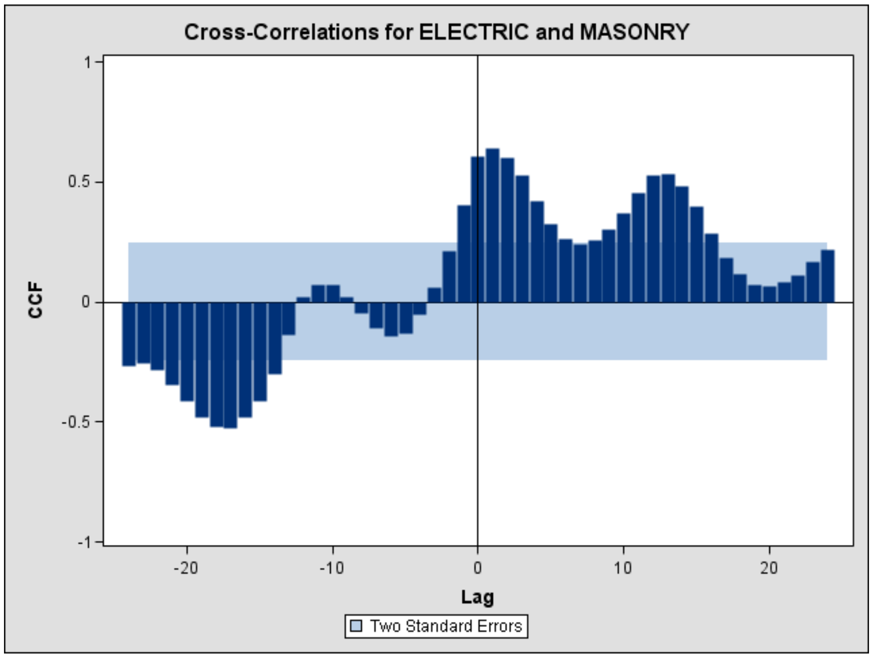
Output 28.3.2. Correlation Panel (Experimental)



Output 28.3.3. Seasonal Decomposition Panel (Experimental)



Output 28.3.4. Cross Correlation Plot (Experimental)



References

Greene, W.H. (1999), *Econometric Analysis*, Fourth Edition, New York: Macmillan.

Hodrick, R. and Prescott, E. (1980), "Post-War U.S. Business Cycles: An Empirical Investigation," Discussion Paper 451, Carnegie Mellon University.

Makridakis, S. and Wheelwright, S.C. (1978), *Interactive Forecasting: Univariate and Multivariate Methods*, Second Edition, San Francisco: Holden-Day, 198-201.

Pyle, D. (1999), *Data Preparation for Data Mining*, San Francisco: Morgan Kaufman Publishers, Inc.

Stoffer, D.S., Toloi, C.M.C. (1992), "A Note on the Ljung-Box-Pierce Portmanteau Statistic with Missing Data," *Statistics and Probability Letters* 13, 391-396.

Wheelwright, S.C. and Makridakis, S. (1973), *Forecasting Methods for Management*, Third Edition, New York: Wiley-Interscience, 123-133.

Chapter 29
The UCM Procedure

Chapter Contents

Chapter 29
The UCM Procedure

Overview

The UCM procedure analyzes and forecasts equally spaced univariate time series data using the Unobserved Components Models (UCM). The UCMs are also called *Structural Models* in the time series literature. A UCM decomposes the response series into components such as trend, seasonals, cycles, and the regression effects due to predictor series. The components in the model are supposed to capture the salient features of the series that are useful in *explaining* and *predicting* its behavior. Harvey (1989) is a good reference for time series modeling using the UCMs. Harvey calls the components in a UCM the “stylized facts” about the series under consideration. Traditionally, the ARIMA models and, to some limited extent, the Exponential Smoothing models, have been the main tools in the analysis of this type of time series data. It is fair to say that the UCMs capture the versatility of the ARIMA models while possessing the interpretability of the smoothing models. A thorough discussion of the correspondence between the ARIMA models and the UCMs, and the relative merits of the UCM and ARIMA modeling is given in Harvey (1989). The UCMs are also very similar to another set of models, called the *Dynamic Models*, that are popular in the Bayesian time series literature (West and Harrison 1999). In SAS/ETS you can use PROC ARIMA for the ARIMA modeling and the Time Series Forecasting System for a point-and-click interface to the ARIMA and exponential smoothing modeling.

You can use the UCM procedure to fit a wide range of UCMs that can incorporate complex trend, seasonal, and cyclical patterns and can include multiple predictors. It provides a variety of diagnostic tools to assess the fitted model and to suggest the possible extensions or modifications. The components in the UCM provide a succinct description of the underlying mechanism governing the series. You can print, save, or plot the estimates of these component series. Along with the standard forecast and residual plots, the study of these component plots is an essential part of the time series analysis using the UCMs. Once a suitable UCM is found for the series under consideration, it can be used for a variety purposes. For example, it can be used for

- forecasting the values of the response series and the component series in the model
- obtaining a model-based seasonal decomposition of the series
- obtaining a “denoised” version and interpolating the missing values of the response series in the historical period
- obtaining the full sample or “smoothed” estimates of the component series in the model

Experimental graphics are now available with the UCM procedure. For more information, see the "ODS Graphics" section on page 1664.

Getting Started

The analysis of time series using the UCMs involves recognizing the salient features present in the series and modeling them suitably. The UCM procedure provides a variety of models for modeling the commonly observed features in time series. These models are discussed in detail later in this section. First the procedure is illustrated using a few examples.

A Seasonal Series with a Linear Trend

The airline passenger series, given as Series G in Box and Jenkins (1976), is often used in time series literature as an example of a nonstationary seasonal time series. This series is a monthly series consisting of the number of airline passengers who traveled during the years 1949 to 1960. Its main features are a continual rise in the number of passengers from year to year and the seasonal variation in the numbers during any given year. It also exhibits an increase in variability around the trend. A log transformation is used to stabilize this variability. These trend and seasonal features of the series are apparent in the following plot (Figure 29.1):

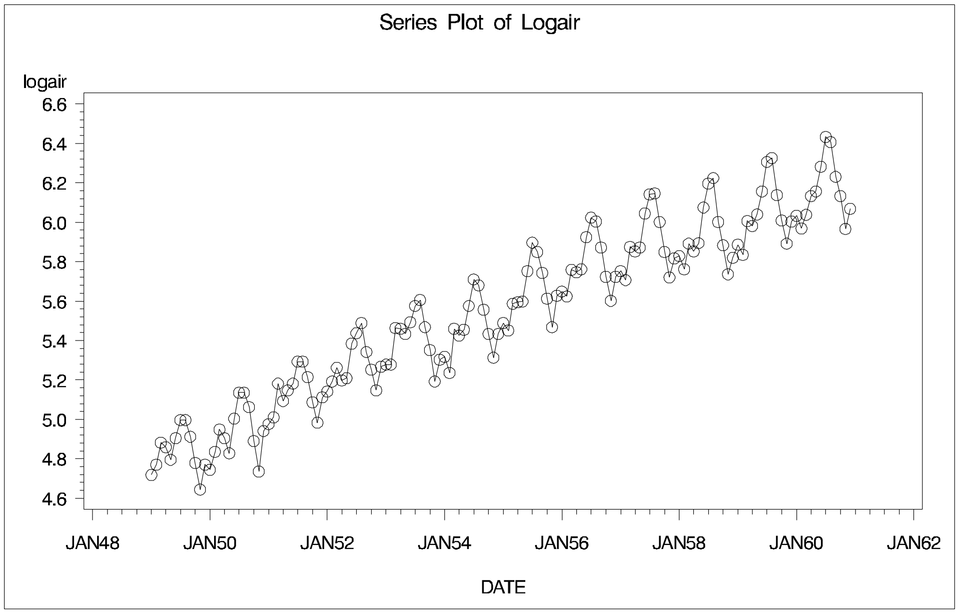


Figure 29.1. Series Plot of Log Transformed Airline Passenger Series

In this example this series will be modeled using an unobserved component model called the Basic Structural Model (BSM). The BSM models a time series as a sum of three stochastic components; a trend component μ_t, a seasonal component γ_t, and random error ϵ_t. Formally, a BSM for a response series y_t can be described as

$$y_t = \mu_t + \gamma_t + \epsilon_t$$

Each of the stochastic components in the model is modeled separately. The random error ϵ_t, also called the *irregular component*, is modeled simply as a sequence of independent, identically distributed (i.i.d.) zero mean Gaussian random variables. The trend and the seasonal components can be modeled in a few different ways. The model for trend used here is called a *locally linear time trend*. This trend model can be written as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad & \eta_t &\sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

These equations specify a trend where the level μ_t as well as the slope β_t is allowed to vary over time. This variation in slope and level is governed by the variances of the disturbance terms η_t and ξ_t in their respective equations. Some interesting special cases of this model arise by manipulating these disturbance variances. For example, if the variance of ξ_t is zero, the slope will be constant (equal to β_0) and, if in addition, the variance of η_t is also zero, μ_t will be a deterministic trend given by the line $\mu_0 + \beta_0 t$. The seasonal model used in this example is called a trigonometric seasonal. The stochastic equations governing a trigonometric seasonal are explained later. However, it is interesting to note here that this seasonal model reduces to the familiar regression with deterministic seasonal dummies if the variance of the disturbance terms in its equations is equal to zero. The following SAS statements specify a BSM with these three components:

```
proc ucm data=series_g;
     id date interval=month;
     model logair;
     irregular;
     level;
     slope;
     season length=12 type=trig print=smooth;
     estimate;
     forecast lead=24 print=decomp;
 run;
```

The PROC statement signifies the start of the UCM procedure and the input data set containing the dependent series is specified there. The optional ID statement is used to specify a date, datetime, or time identification variable, *date* in this example, to label the observations. The INTERVAL=MONTH option in the ID statement indicates that the measurements were collected on a monthly basis. The model specification begins with the MODEL statement, where the dependent series is specified (*logair* in this case). After this the components in the model are specified using separate statements that enable controlling their individual properties. The IRREGULAR statement is used to specify the irregular component ϵ_t, and the trend component μ_t is specified using the LEVEL and SLOPE statements. The seasonal component γ_t is specified using the SEASON statement. The specifics of the seasonal characteristics such as its season length, its stochastic evolution properties, etc, are specified using the options in the SEASON statement. The seasonal used in this example has season length 12, corresponding to the monthly seasonality, and is of the *trigonometric* type (different types of seasonals are explained later in this section). The parameters of this model are the variances of the disturbance terms in the evolution equations of μ_t, β_t and γ_t and the variance of the *irregular* component ϵ_t. These parameters are estimated by maximizing the likelihood of the data. The ESTIMATE statement options can be used to specify the span of data used in parameter estimation and to display and save the results of the estimation step and the model diagnostics. You can use the estimated model to obtain the forecasts of the series as well as the components. The options in the individual component statements can be used to display the component forecasts, for example, PRINT=SMOOTH option in the SEASON statement requests the displaying of smoothed forecasts of the seasonal component γ_t. The series forecasts and forecasts of the sum of components can be requested using the FORECAST statement. The option PRINT=DECOMP in the FORECAST statement requests the printing of the smoothed trend μ_t and the trend plus seasonal ($\mu_t + \gamma_t$).

The parameter estimates for this model are displayed in Figure 29.2.

```
                                The UCM Procedure

                       Final Estimates of the Free Parameters

                                                       Approx                 Approx
Component    Parameter          Estimate    Std Error    t Value    Pr > |t|

Irregular    Error Variance     0.00023436    0.0001079       2.17      0.0298
Level        Error Variance     0.00029828    0.0001057       2.82      0.0048
Slope        Error Variance     9.8572E-13   6.7141E-10       0.00      0.9988
Season       Error Variance     0.00000356   1.32347E-6       2.69      0.0072
```

Figure 29.2. BSM for the Logair Series

The estimates suggest that except for the slope component the disturbance variances of all the components are significant, that is, all these components are stochastic. The slope component, however, appears to be deterministic because its error variance is quite insignificant. It may then be useful to check if the slope component can be dropped from the model, that is if $\beta_0 = 0$. This can be checked by examining the significance analysis table of the components given in Figure 29.3.

```
                    Significance Analysis of Components
                         (Based on the Final State)

                Component         DF     Chi-Square     Pr > ChiSq

                Irregular          1           0.08         0.7747
                Level              1         117867         <.0001
                Slope              1          43.78         <.0001
                Season            11         507.75         <.0001
```

Figure 29.3. Component Significance Analysis for the Logair Series

This table provides the significance of the components in the model at the end of the estimation span. If a component is deterministic, this analysis is equivalent to checking whether the corresponding regression effect is significant. However, if a component is stochastic then this analysis only pertains to the portion of the series near the end of the estimation span. In this example the slope appears quite significant and should be retained in the model, possibly as a deterministic component. Note that, on the basis of this table, the irregular component's contribution appears insignificant towards the end of the estimation span, however, since it is a *stochastic* component it cannot be dropped from the model on the basis of this analysis alone. The slope component can be made deterministic by holding the value of its error variance fixed at zero. This is done by modifying the SLOPE statement as follows:

```
   slope variance=0 noest;
```

After a tentative model is fit its adequacy can be checked by examining different Goodness of Fit measures and other diagnostic tests and plots that are based on the model residuals. The table given in Figure 29.4 shows the goodness fit statistics that are computed using the one step ahead prediction errors (see "Statistics of Fit"). These measures indicate a good agreement between the model and the data. Additional diagnostics measures are also printed by default but are not shown here.

```
                              The UCM Procedure

                       Fit Statistics Based on Residuals

                 Mean Squared Error                     0.00147
                 Root Mean Squared Error                0.03830
                 Mean Absolute Percentage Error         0.54132
                 Maximum Percent Error                  2.19097
                 R-Square                               0.99061
                 Adjusted R-Square                      0.99046
                 Random Walk R-Square                   0.99220
                 Amemiya's Adjusted R-Square            0.99017

                   Number of non-missing residuals used for
                      computing the fit statistics = 131
```

Figure 29.4. Fit Statistics for the Logair Series

Once the model appears satisfactory, it can be used for forecasting. An interesting feature of the UCM procedure is that, apart from the series forecasts, you can request the forecasts of the individual components in the model. The plots of component forecasts can be useful in understanding their contributions to the series. In what follows, a few such plots are shown. The first plot (Figure 29.5) shows the smoothed trend of the series.

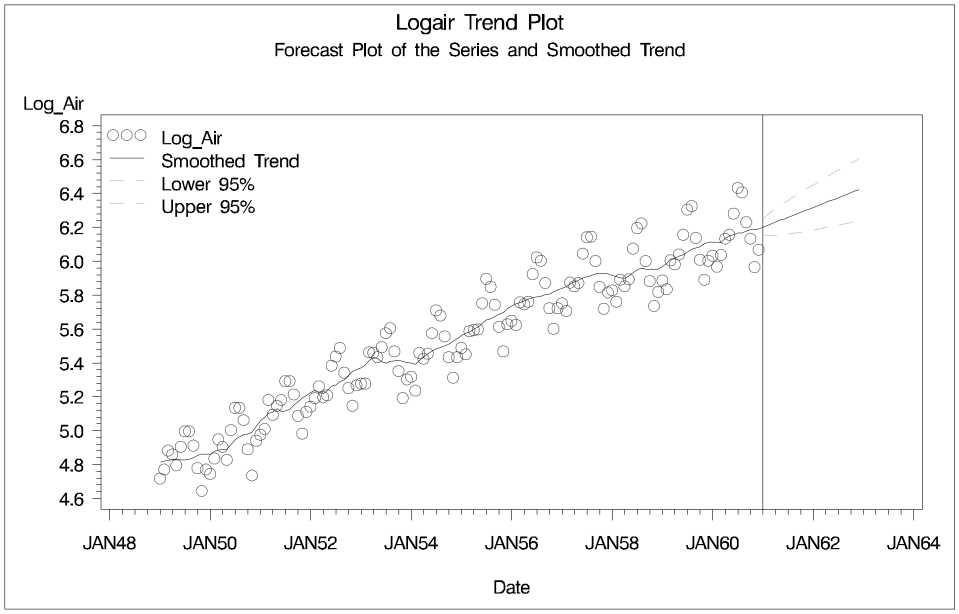


Figure 29.5. Smoothed Trend in the Logair Series

The second plot (Figure 29.6) shows the seasonal component by itself.

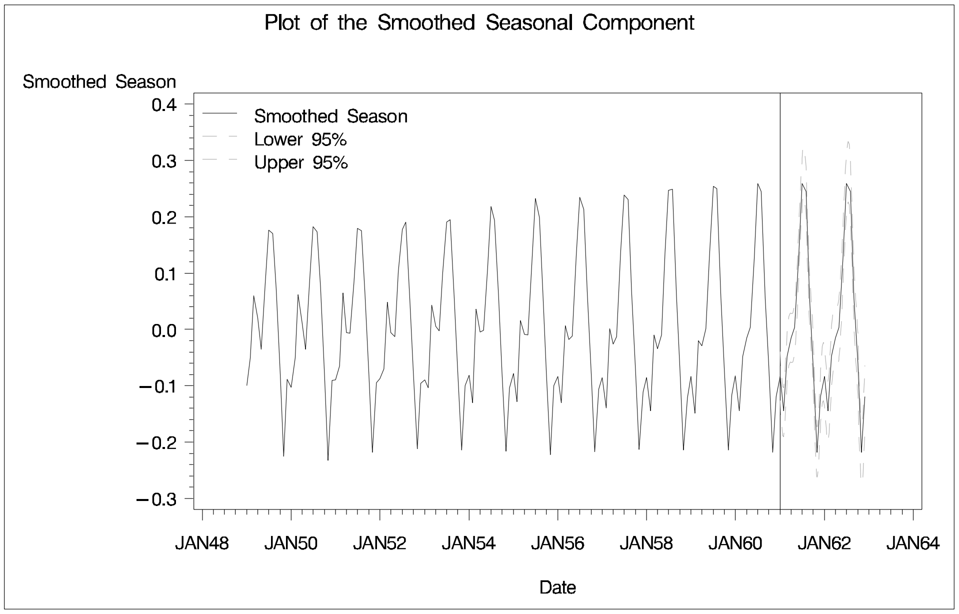


Figure 29.6. Smoothed Seasonal in the Logair Series

The plot of the sum of trend and seasonal is shown in Figure 29.7. You can see that, at least visually, the model seems to fit the data well. In all these decomposition plots the component estimates are extrapolated for two years in the future.

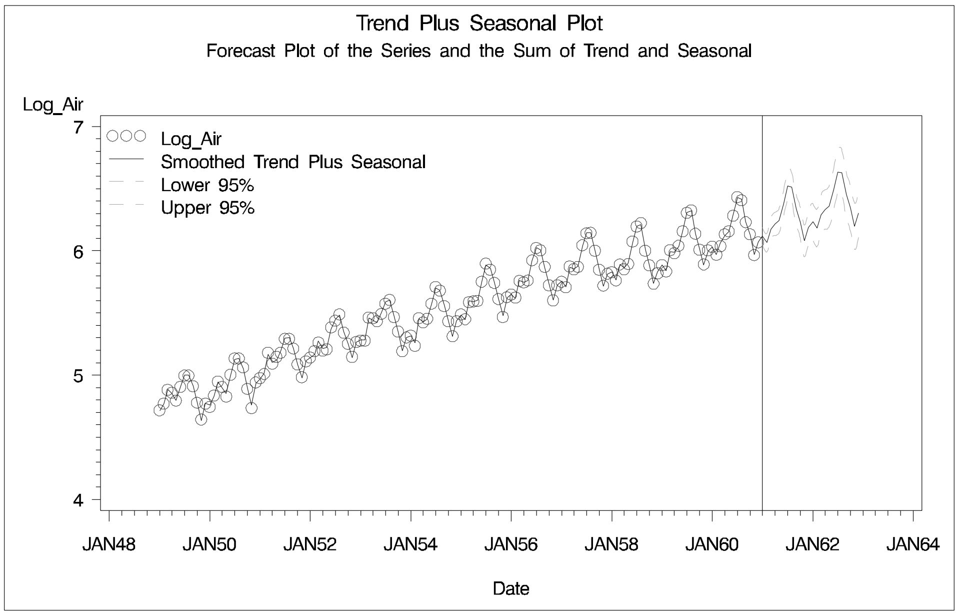


Figure 29.7. Smoothed Trend plus Seasonal in the Logair Series

A Series with Cyclical Component

In this example another well known series, Wolfer's sunspot data (Anderson 1971), is considered. The data consist of yearly sunspot numbers recorded from 1749 to 1924. These sunspot numbers are known to have a cyclical pattern with period of about eleven years. A time series plot of this series is given in Figure 29.8. From the plot it is difficult to discern any other specific pattern to these numbers.

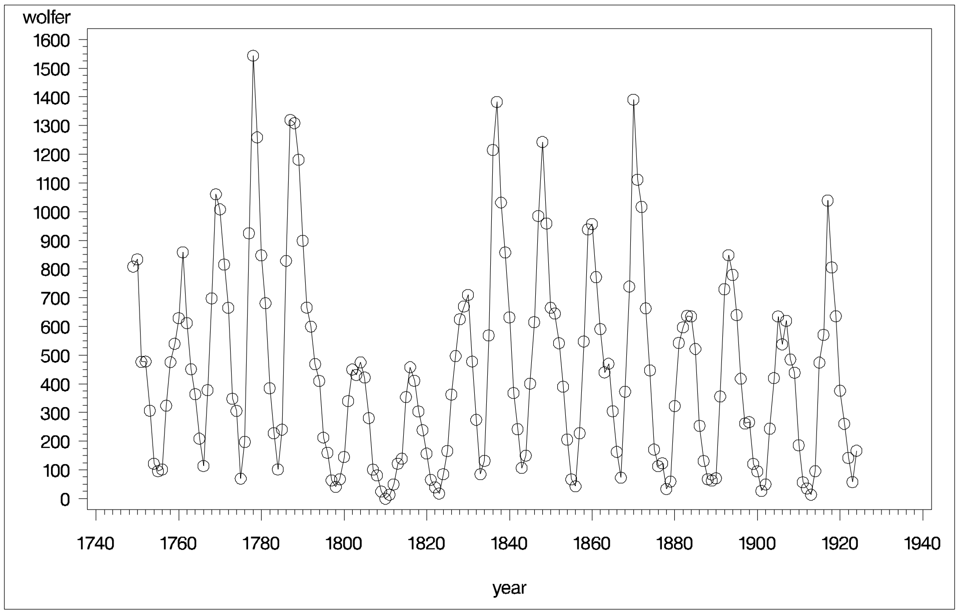


Figure 29.8. Wolfer Sunspot Numbers

The following syntax specifies a UCM that includes a cycle component and a level component.

```
proc ucm data=sunspot;
     id year interval=year;
     model wolfer;
     irregular;
     level;
     cycle print=smooth;
     estimate;
     forecast lead=12 print=decomp;
 run;
```

In this model the trend of the series is modeled as a time-varying level component without any persistent upward or downward drift, that is, no slope component is included. The cyclical behavior of the series is modeled using a cycle component that is a damped sinusoidal with fixed period and time varying amplitude. The parameters of the cycle are its period, the damping factor, and the variance of the disturbance terms in its stochastic equations. They are estimated from the data. In this case the estimate of the cycle period turns out to be approximately 10.58 years, consistent with the known results. As in the earlier example, it is informative to see the decomposition plots of the series. The first plot (Figure 29.9) shows the smoothed trend of the series.

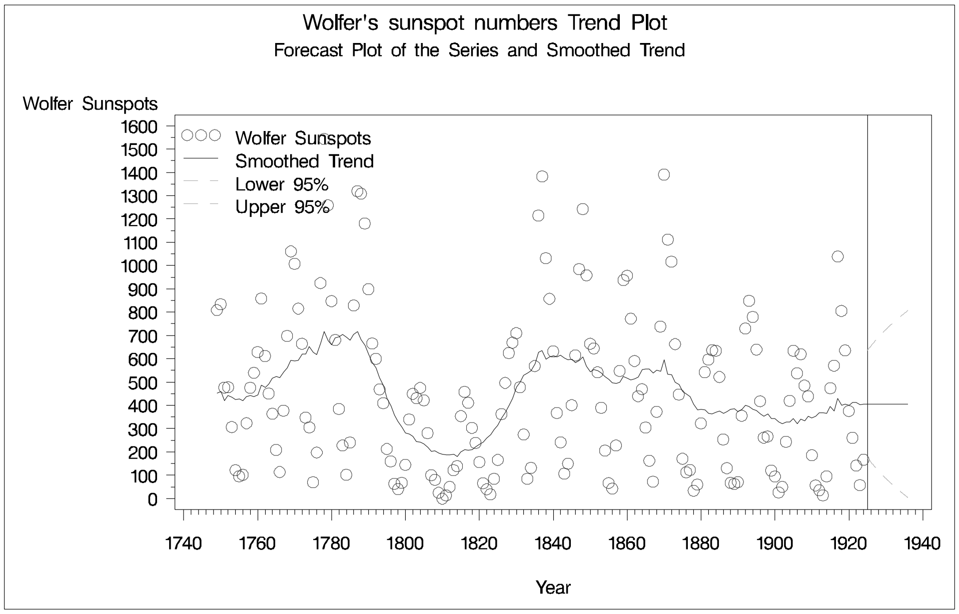


Figure 29.9. Smoothed Trend

The second plot shows the cycle component (Figure 29.10).

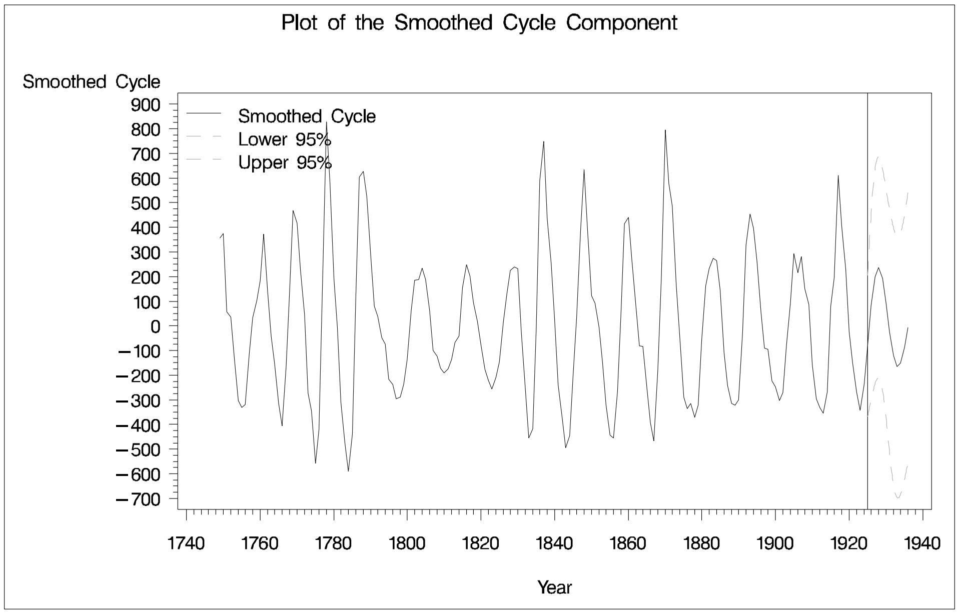


Figure 29.10. Smoothed Cycle

The plot of sum of trend and cycle is shown in Figure 29.11.

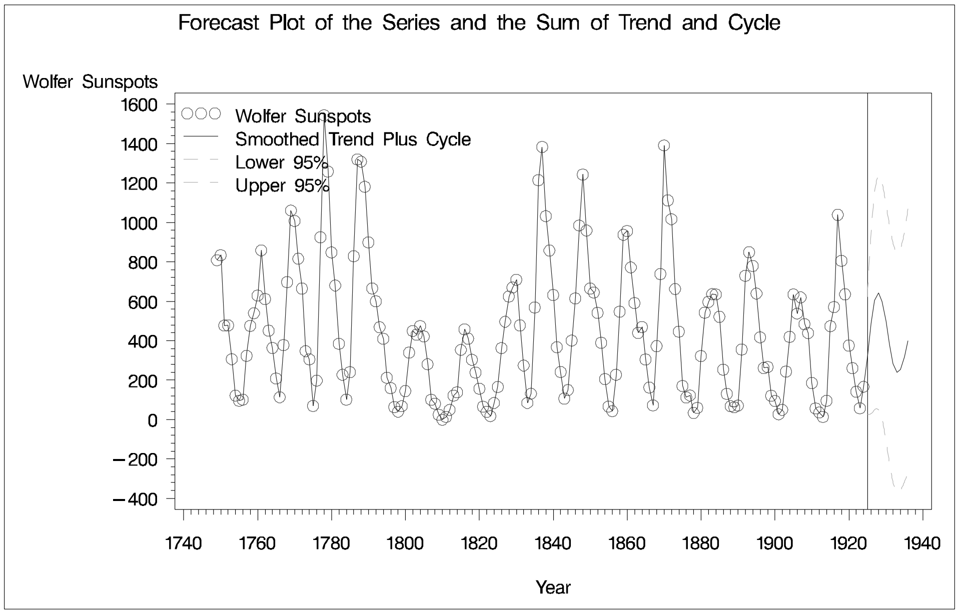


Figure 29.11. Plot of Smoothed Trend Plus Cycle

A Series with Level Shift

In this example the series consists of the yearly level readings of the river Nile recorded at Aswan (see Cobb 1978). The data consists of readings from the years 1871 to 1970. The series does not show any apparent trend or any other distinctive patterns; however, there is a shift in the level starting at the year 1899. This shift could be attributed to the start of construction of a dam near Aswan in that year. A time series plot of this series is given in Figure 29.12.

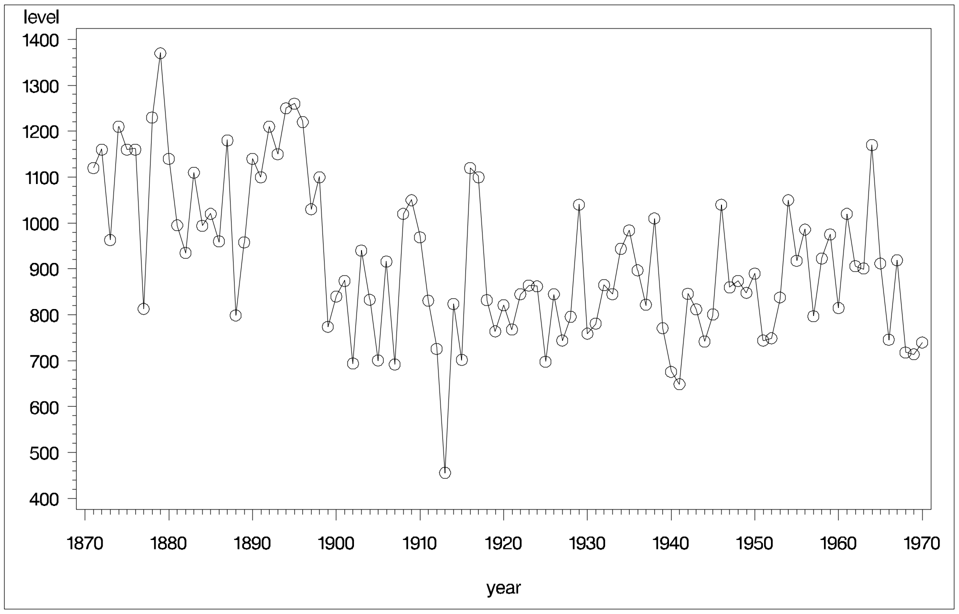


Figure 29.12. Nile River Level

The following syntax specifies a UCM that models the level of the river as a locally constant series with a shift in the year 1899, represented by a dummy regressor (Shift1899).

```
proc ucm data=nile;
   id year interval=year;
   model nile_level = shift1899;
   irregular;
   level;
   estimate;
   forecast print=decomp;
run;
```

The decomposition plots of this model can be easily obtained. However, it is instructive to see the plot of the smoothed trend obtained without using this regressor in the model. This plot is given in Figure 29.13. The plot shows a noticeable drop in the smoothed river level around 1899.

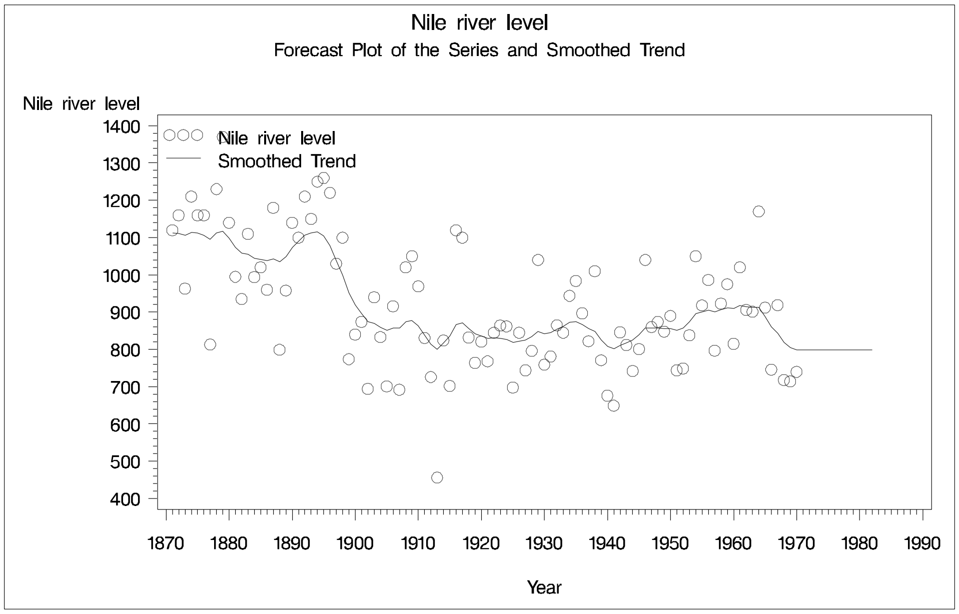


Figure 29.13. Smoothed Trend without the Shift of 1899

The second plot shows the smoothed trend including the correction due to the shift in the year 1899 (Figure 29.14). Notice the simplicity in the shape of the smoothed curve after the incorporation of the shift information.

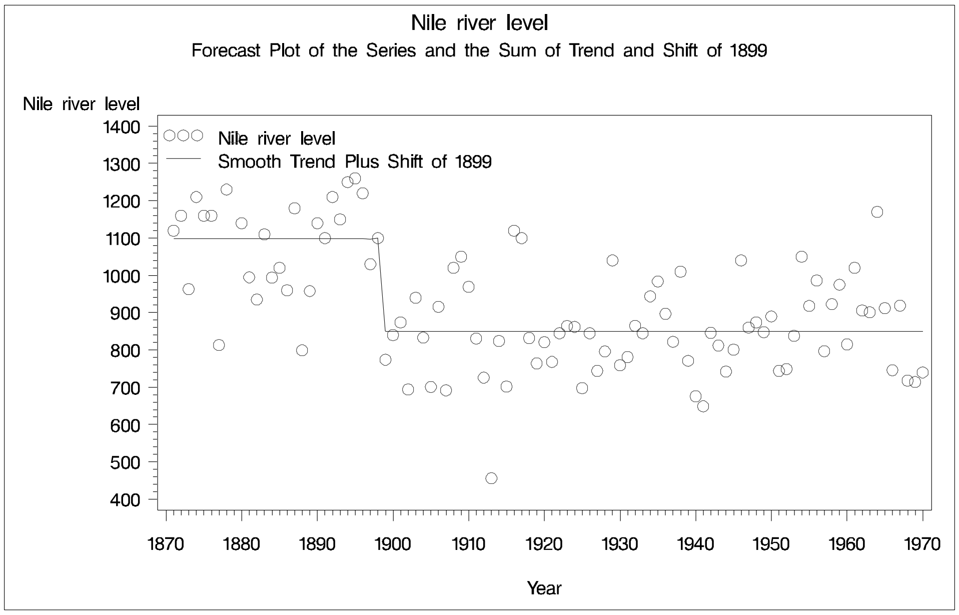


Figure 29.14. Smoothed Trend Plus Shift of 1899

An Introduction to Unobserved Component Models

A general UCM considered in this procedure can be described as

$$\begin{aligned} y_t &= \mu_t + \gamma_t + \psi_t + r_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{m} \beta_j x_{jt} + \epsilon_t \\ \epsilon_t &\sim i.i.d.\ N(0, \sigma_\epsilon^2) \end{aligned}$$

The terms μ_t, γ_t, ψ_t, and r_t represent the trend, seasonal, cyclical and the autoregressive components, respectively. In fact the model can contain multiple seasonals and cycles, and the seasonals can be of different types. For simplicity of discussion the above model contains only one of each of these components. The regression term, $\sum_{j=1}^{m} \beta_j x_{jt}$, includes variables with values supplied in the input dataset. The $\sum_{i=1}^{p} \phi_i y_{t-i}$ is a regression term involving the lags of the dependent variable. It is written separately because its mathematical treatment is slightly different (see "Details"). The disturbance term ϵ_t, also called the $irregular$ component, is assumed to be a Gaussian white noise with variance σ_ϵ^2. By controlling the presence or absence of various terms and by choosing the proper flavor of the included terms, the UCMs can generate a rich variety of time series patterns. A UCM can be applied to variables after transforming them by transforms such as *log* and *difference*.

The components μ_t, γ_t, ψ_t, and r_t model structurally different aspects of the time series. For example, the trend μ_t models the natural tendency of the series in the absence of any other perturbing effects such as seasonality, cyclical components, and the effects of exogenous variables while the seasonal component γ_t models the correction to the level due to the seasonal effects. These components are assumed to be statistically independent of each other and independent of the irregular component. All of the component models can be thought of as stochastic generalizations of the relevant deterministic patterns in time. This way the deterministic cases emerge as special cases of the stochastic models. The different models available for these unobserved components are discussed next.

Modeling the Trend

As mentioned earlier, the trend in a series can be loosely defined as the natural tendency of the series in the absence of any other perturbing effects. The UCM procedure offers two ways to model the trend component μ_t. The first model, called the Random Walk (RW) model, implies that the trend remains roughly constant throughout the life of the series without any persistent upward or downward drift. In the second model the trend is modeled as a locally linear time trend (LLT). The RW model can be described as

$$\mu_t = \mu_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2)$$

Note that if $\sigma_\eta^2 = 0$ then the model becomes $\mu_t = constant$. In the LLT model the trend is locally linear, consisting of both the *level* and *slope*. The model for μ_t is

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, \qquad \xi_t \sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

The disturbances η_t and ξ_t are assumed to be independent. There are some interesting special cases of this model obtained by setting one or both of the disturbance variances σ_η^2 and σ_ξ^2 equal to zero. If σ_ξ^2 is set equal to zero then you get a linear trend model with fixed slope. If σ_η^2 is set to zero then the resulting model usually has a smoother trend. If both the variances are set to zero then the resulting model is the deterministic linear time trend: $\mu_t = \mu_0 + \beta_0 t$.

These trend patterns can be incorporated in your model using the LEVEL and SLOPE statements in PROC UCM.

Modeling a Cycle

A deterministic cycle ψ_t with frequency λ, $0 < \lambda < \pi$, can be written as

$$\psi_t = \alpha \cos(\lambda t) + \beta \sin(\lambda t)$$

If the argument t is measured on a continuous scale, ψ_t is a periodic function with period $2\pi/\lambda$, amplitude $(\alpha^2 + \beta^2)^{1/2}$, and phase $\tan^{-1}(\beta/\alpha)$. However, if ψ_t is measured only at the integer values it is not exactly periodic, unless $\lambda = (2\pi j)/k$ for some integers j and k. The cycles in their pure form are not used very often in practice. However, they are very useful as building blocks for more complex periodic patterns. It is well known that the periodic pattern of any complexity can be written as a sum of pure cycles of different frequencies and amplitudes. In time series situations it is useful to generalize this simple cyclical pattern to a stochastic cycle that has a fixed period but time varying amplitude and phase. The stochastic cycle considered here is motivated by the following recursive formula for computing ψ_t:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix}$$

starting with $\psi_0 = \alpha$ and $\psi_0^* = \beta$. Note that ψ_t and ψ_t^* satisfy the relation

$$\psi_t^2 + \psi_t^{*2} = \alpha^2 + \beta^2 \quad \text{for all } t$$

A stochastic generalization of the cycle ψ_t can be obtained by adding random noise to this recursion and by introducing a damping factor, ρ, for additional modeling flexibility. This model can be described as follows:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where $0 \le \rho \le 1$, and the disturbances ν_t and ν_t^* are independent $N(0, \sigma_\nu^2)$ variables. The resulting stochastic cycle has a fixed period but time varying amplitude and phase. The stationarity properties of the random sequence ψ_t depend on the damping factor ρ. If $\rho < 1$, ψ_t has a stationary distribution with mean zero and variance $\sigma_\nu^2/(1-\rho^2)$. If $\rho = 1$, ψ_t is non-stationary.

A cycle is incorporated in a UCM by using a CYCLE statement in PROC UCM. Multiple cycles can be included in the model using separate CYCLE statements for each included cycle.

As mentioned before, the cycles are very useful as building blocks for constructing more complex periodic patterns. Periodic patterns of almost any complexity can be created by superimposing cycles of different periods and amplitudes. In particular, the seasonal patterns, general periodic patterns with integer periods, can be constructed as sums of cycles. This important topic of modeling the seasonals is considered next.

Modeling a Seasonal

The seasonal fluctuations are a common source of variation in the time series data. These fluctuations arise because of the regular changes in seasons or some other periodic events. The seasonal effects are regarded as corrections to the general trend of the series due to the seasonal variations, and these effects sum to zero when summed over the full season cycle. Therefore the seasonal component γ_t is modeled as a stochastic periodic pattern of an integer period s such that the sum $\sum_{i=0}^{s-1} \gamma_{t-i}$ is always zero in the mean. The period s is called the season length. Two different models for the seasonal component are considered here. The first model is called the *dummy* variable form of the seasonal component. It is described by the equation

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \qquad \omega_t \sim i.i.d.\ N(0, \sigma_\omega^2)$$

The other type of model is called the *trigonometric* form of the seasonal component. In this case γ_t is modeled as a sum of cycles of different frequencies. This model is given as follows:

$$\gamma_t = \sum_{j=1}^{[s/2]} \gamma_{j,t}$$

where $[s/2]$ equals $s/2$ if s is even and equals $(s-1)/2$ if it is odd. The cycles $\gamma_{j,t}$ have frequencies $\lambda_j = 2\pi j/s$ and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma_{j,t}^* \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma_{j,t-1}^* \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega_{j,t}^* \end{bmatrix}$$

where the disturbances $\omega_{j,t}$ and $\omega_{j,t}^*$ are assumed to be independent and, for fixed j, $\omega_{j,t}$ and $\omega_{j,t}^* \sim N(0, \sigma_\omega^2)$. If s is even then the equation for $\gamma_{s/2,t}^*$ is not needed and $\gamma_{s/2,t}$ is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

The cycles $\gamma_{j,t}$ are called *harmonics*. If the seasonal is deterministic, the decomposition of the seasonal effects into these harmonics is identical to its Fourier decomposition. In this case the sum of squares of the seasonal factors equals the sum of squares of the amplitudes of these harmonics. In many practical situations, the contribution of the high frequency harmonics is negligible and can be ignored, giving rise to a simpler description of the seasonal. In the case of stochastic seasonals the situation may not be so transparent; however, similar considerations still apply. Note that, if the disturbance variance $\sigma_\omega^2 = 0$ then both the dummy and the trigonometric forms of seasonal components reduce to constant seasonal effects. That is, the seasonal component reduces to a deterministic function that is completely determined by its first $s - 1$ values.

The dummy and the trigonometric type seasonals defined above can be considered as *saturated* seasonals that put no restrictions on the $s - 1$ seasonal values. In some cases a more parsimonious representation of the seasonal may be more appropriate. This is particularly useful for seasonals with large season lengths. Parsimonious representations of the seasonals can be obtained in several different ways. Some of these ways involve the reuse of the already introduced cycle and seasonal components. One possibility is to consider special cases of the trigonometric seasonals obtained by deleting a few of the $[s/2]$ harmonics used in the sum. For example, a slightly smoother seasonal of length 12, corresponding to the monthly seasonality, can be obtained by deleting the highest frequency harmonic of period 2. That is, such a seasonal will be a sum of five stochastic cycles that have periods 12, 6, 4, 3, and 2.4. Another possibility is to consider a seasonal of a large season length as a sum of two or more seasonals that are each of much smaller season lengths. One more possibility is to restrict the seasonal values within certain blocks to be the same. An example of such a situation is as follows: Consider an hourly series that may show periodic variation that is attributable to the day of the week, and to the hour of the day. The hour of the day effect can be modeled as a simple saturated seasonal of season length 24. The day of the week effect could be modeled as a seasonal of season length 168 that restricts the seasonal values within a given day to be equal. Such a seasonal could be called a block-seasonal of season length 7 and block length 24.

These different types of seasonal patterns can be included in a UCM using a combination of SEASON, BLOCKSEASON, and CYCLE statements in PROC UCM.

Modeling the Autoregression

An autoregression of order one can be thought of as a special case of a cycle when the frequency λ is either 0 or π. Modeling this special case separately helps interpretation and parameter estimation. The auto-regression component r_t is modeled as follows:

$$r_t = \rho r_{t-1} + \nu_t, \quad \nu_t \ \sim \ i.i.d. \ \ N(0, \sigma_\nu^2)$$

where $-1 \leq \rho < 1$.

The Regression Terms

The regression terms $\sum_{j=1}^{m} \beta_j x_{jt}$ and $\sum_{i=1}^{p} \phi_i y_{t-i}$ supply additional flexibility to the model. Lags, differences, and other transformations can be applied to the variables.

The Model Parameters

The parameter vector in a UCM consists of the variances of the disturbance terms of the unobserved components, the damping coefficients and frequencies in the cycles, the damping coefficient in the autoregression, and the regression coefficients in the regression terms. These parameters are estimated by maximizing the likelihood. It is possible to restrict the values of the model parameters to user specified values.

Model Specification

A UCM is specified by describing the components in the model. For example, consider the model

$$y_t = \mu_t + \gamma_t + \epsilon_t$$

consisting of the irregular, level, slope, and the seasonal components. This model is called the Basic Structural Model (BSM) by Harvey. The syntax for a BSM with monthly seasonality of trigonometric type is

```
model y;
   irregular;
   level;
   slope;
   season length=12 type=trig;
```

Similarly the syntax

```
model y = x;
   irregular;
   level;
   slope variance=0 noest;
   season length=12 type=dummy;
```

specifies a BSM with dependent variable y, a regressor x and dummy type monthly seasonality. Moreover, the disturbance variance of the slope component is restricted to zero, giving rise to a local linear trend with fixed slope.

A model can contain multiple cycle and seasonal components. In such cases the model syntax contains a separate statement for each of these multiple cycle or seasonal components; for example, the syntax for model containing irregular and level components along with two cycle components could be as follows:

```
model y = x;
   irregular;
   level;
   cycle;
   cycle;
```

Syntax

Syntax

The UCM procedure uses the following statements.

PROC UCM *options*;
  BY *variables*;
  ID *variable options*;
  MODEL *dependent variable < = regressors >* ;
  IRREGULAR *options*;
  LEVEL *options*;
  SLOPE *options*;
  SEASON *options*;
  BLOCKSEASON *options*;
  CYCLE *options*;
  AUTOREG *options*;
  DEPLAG *options*;
  ESTIMATE *options*;
  FORECAST *options*;
  NLOPTIONS *options*;

Functional Summary

The statements and options controlling the UCM procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set	PROC UCM	DATA=
write parameter estimates to an output data set	ESTIMATE	OUTEST=
write forecasts and smoothed values of the response series and components to an output data set	FORECAST	OUTFOR=
BY Groups		
specify BY-group processing	BY	
ID Variable		
specify a date or time identification variable	ID	
specify the time interval between observations	ID	INTERVAL=
control the alignment of SAS Date values	ID	ALIGN=
Options for Specifying the Model		

Description	Statement	Option
specify the response series and, optionally, the predictor series	MODEL	
specify the initial value for the disturbance variance of the irregular component	IRREGULAR	VARIANCE=
fix the value of the disturbance variance of the irregular component to the specified initial value	IRREGULAR	NOEST
specify the initial value for the disturbance variance of the level component	LEVEL	VARIANCE=
fix the value of the disturbance variance of the level component to the specified initial value	LEVEL	NOEST
specify the initial value for the disturbance variance of the slope component	SLOPE	VARIANCE=
fix the value of the disturbance variance of the slope component to the specified initial value	SLOPE	NOEST
specify the season length of a seasonal component	SEASON	LENGTH=
specify the type of a seasonal component	SEASON	TYPE=
specify the initial value for the disturbance variance of a seasonal component	SEASON	VARIANCE=
fix the value of the disturbance variance of the seasonal component to the specified initial value	SEASON	NOEST
specify the block size of a block seasonal component	BLOCKSEASON	BLOCKSIZE=
specify the number of blocks of a block seasonal component	BLOCKSEASON	NBLOCKS=
specify the relative position of the first observation within the block of a block seasonal component	BLOCKSEASON	OFFSET=
specify the initial value for the disturbance variance of a block seasonal component	BLOCKSEASON	VARIANCE=
fix the value of the disturbance variance of the block seasonal component to the specified initial value	BLOCKSEASON	NOEST
specify the initial value for the period of a cycle component	CYCLE	PERIOD=
specify the initial value for the damping factor of a cycle component	CYCLE	RHO=
specify the initial value for the disturbance variance of the cycle component	CYCLE	VARIANCE=
fix the values of the parameters of the cycle component to the specified initial values	CYCLE	NOEST=
specify the initial value for the damping factor of the autoreg component	AUTOREG	RHO=
specify the initial value for the disturbance variance of the autoreg component	AUTOREG	VARIANCE=
fix the values of the parameters of the autoreg component to the specified initial values	AUTOREG	NOEST=

Description	**Statement**	**Option**
specify the lags of the response series to be included in the model	DEPLAG	LAGS=
specify the initial values for the lag coefficients for the response lags	DEPLAG	PHI=
fix the values of lag coefficients to the specified initial values	DEPLAG	NOEST
Options to Control the Nonlinear Optimization in Estimation Process		
specify an optimization algorithm	NLOPTIONS	TECH=
limit number of iterations during the optimization	NLOPTIONS	MAXITER=
limit number of function evaluations	NLOPTIONS	MAXFUNC=
specify function convergence criteria	NLOPTIONS	ABSFTOL=
specify gradient convergence criteria	NLOPTIONS	ABSGTOL=
specify parameter convergence criteria	NLOPTIONS	ABSXTOL=
Options to Control the Observation Span in Estimation and Forecasting		
specify how many starting response series measurements to exclude during the model estimation phase	ESTIMATE	SKIPFIRST=
specify how many ending response series measurements to exclude during the model estimation phase	ESTIMATE	BACK=
specify how many starting response series measurements to exclude during the forecasting phase	FORECAST	SKIPFIRST=
specify how many ending response series measurements to exclude during the forecasting phase	FORECAST	BACK=
specify how many periods to forecast beyond the forecast span	FORECAST	LEAD=
specify size of forecast confidence limits	FORECAST	ALPHA=
Options to Control the Printing		
suppress printing altogether	PROC UCM	NOPRINT
turn all the print options on	PROC UCM	PRINTALL
suppress the printing of parameter estimates, goodness of fit statistics, and other estimation output	ESTIMATE	PRINT=
suppress printing of forecast output	FORECAST	PRINT=
print filtered or smoothed estimate of the irregular component	IRREGULAR	PRINT=
print filtered or smoothed estimate of the level component	LEVEL	PRINT=
print filtered or smoothed estimate of the slope component	SLOPE	PRINT=
print filtered or smoothed estimate of the autoreg component	AUTOREG	PRINT=
print filtered or smoothed estimate of a cycle component	CYCLE	PRINT=

Description	**Statement**	**Option**
print filtered or smoothed estimate of a seasonal component	SEASON	PRINT=
print filtered or smoothed estimate of a block seasonal component	BLOCKSEASON	PRINT=
print parameter estimation related information	ESTIMATE	PRINT=
print forecasts, and smoothed estimates of model decomposition	FORECAST	PRINT=

PROC UCM Statement

PROC UCM *options;*

The following options can be used in the PROC UCM statement:

DATA= *SAS-data-set*
specifies the name of the SAS data set containing the time series. If the DATA= option is not specified in PROC UCM statement, the most recently created SAS data set is used.

NOPRINT
turns off all the printing for the procedure. The subsequent print options in the procedure are ignored.

PRINTALL
turns on all the printing options for the procedure. The subsequent noprint options in the procedure are ignored.

BY Statement

BY *variables;*

A BY statement can be used in the UCM procedure to process a data set in groups of observations defined by the BY variables. The model specified using the MODEL and other component statements is applied to all the groups defined by the BY variables.

ID Statement

ID *variable* **INTERVAL=** *value* < **ALIGN=** *value* > **;**

The ID statement specifies a variable that identifies observations in the input data set. The variable specified in the ID statement is included in the OUT= data set. Note that the ID *variable* is usually a SAS date, time, or date-time variable. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= option.

ALIGN= *value*
controls the alignment of SAS dates used to identify output observations. The

ALIGN= option has the following possible values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING. The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

INTERVAL= *value*

specifies the time interval between observations. This option is required in the ID statement. The INTERVAL=*value* is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data.

MODEL Statement

MODEL *dependent* < = *regressors* > **;**

The MODEL statement specifies the response variable and, optionally, the predictor variables for the UCM model. This is a required statement in the procedure.

IRREGULAR Statement

IRREGULAR < *options* > **;**

The IRREGULAR statement is used to include an *irregular* component in the model. There can be at most one IRREGULAR statement in the model specification. The irregular component corresponds to the overall random error, ϵ_t, in the model; it is modeled as a sequence of independent, zero mean, Gaussian random variables with variance σ_ϵ^2. The options in this statement enable you to specify the value of σ_ϵ^2 and to output the forecasts of ϵ_t. As a default, σ_ϵ^2 is estimated using the data and the component forecasts are not saved or displayed. A few examples of the IRREGULAR statement are given next. In the first example the statement is in its simplest form, resulting in the inclusion of an *irregular* component with unknown variance.

```
irregular;
```

The following statement provides a starting value for σ_ϵ^2, to be used in the non-linear parameter estimation process. It also requests the printing of smoothed predictions of ϵ_t. The smoothed irregulars are useful in model diagnostics.

```
irregular variance=4 print=smooth;
```

NOEST

This option fixes the value of σ_ϵ^2 to the value specified in the VARIANCE= option.

PRINT= FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)

This option requests printing of filtered or smoothed estimate of the irregular component.

VARIANCE= *value*

This option is used to supply an initial value for σ_ϵ^2 during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

LEVEL Statement

LEVEL < *options* > **;**

The LEVEL statement is used to include a *level* component in the model. The level component, either by itself or, together with a *slope* component, form the *trend* component, μ_t, of the model. If the slope component is absent, the resulting trend is a Random Walk (RW) specified by the following equations:

$$\mu_t = \mu_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2)$$

If the slope component is present, signified by the presence of a SLOPE statement that is explained later, a Locally Linear Trend (LLT) is obtained. The equations of LLT are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad & \eta_t &\sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

In either case, the options in the LEVEL statement are used to specify the value of σ_η^2 and to request forecasts of μ_t. The SLOPE statement is used for similar purposes in the case of slope β_t. The following examples illustrate the use of LEVEL statement. Assuming that a SLOPE statement is not added subsequently, a simple Random Walk trend is specified by the following statement:

```
level;
```

The following statements specify a locally linear trend with value of σ_η^2 fixed at 4. It also requests printing of filtered values of μ_t. The value of σ_ξ^2, the disturbance variance in the slope equation, will be estimated from the data.

```
level variance=4 noest print=filter;
slope;
```

NOEST
This option fixes the value of σ_η^2 to the value specified in the VARIANCE= option.

PRINT=FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)
This option requests printing of filtered or smoothed estimate of the level component.

VARIANCE= *value*
This option is used to supply an initial value for σ_η^2, the disturbance variance in the μ_t equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SLOPE Statement

SLOPE < *options* > **;**

The SLOPE statement is used to include a *slope* component in the model. The slope component cannot be used without the level component. The level and slope specifications jointly define the trend component of the model. A SLOPE statement without the accompanying LEVEL statement will be ignored. The equations of the trend, defined jointly by the level μ_t and slope β_t, are as follows:

$$\begin{aligned}\mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma^2_\eta) \\ \beta_t &= \beta_{t-1} + \xi_t, \qquad\qquad \xi_t \sim i.i.d.\ N(0, \sigma^2_\xi)\end{aligned}$$

The SLOPE statement is used to specify the value of the disturbance variance, σ^2_ξ, in the slope equation, and to request forecasts of β_t. The following examples illustrate this statement:

```
level;
slope;
```

These statements request including a locally linear trend in the model. The disturbance variances σ^2_η and σ^2_ξ will be estimated from the data. You can request a locally linear trend with fixed slope using the following statements:

```
level;
slope variance=0 noest;
```

NOEST

This option fixes the value of the disturbance variance, σ^2_ξ, to the value specified in the VARIANCE= option.

PRINT=FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)

This option requests printing of filtered or smoothed estimate of the slope component β_t.

VARIANCE= *value*

This option is used to supply an initial value for the disturbance variance, σ^2_ξ, in the β_t equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SEASON Statement

SEASON LENGTH= *integer* < *options* > **;**

The SEASON or the SEASONAL statement is used to specify a *seasonal* component, γ_t, in the model. A seasonal can be one of the two types, DUMMY or TRIGONOMETRIC. A DUMMY type seasonal with season length s satisfies the following stochastic equation:

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \qquad \omega_t \sim i.i.d.\ N(0, \sigma_\omega^2)$$

The equations for a TRIGONOMETRIC type seasonal are as follows:

$$\gamma_t = \sum_{j=1}^{[s/2]} \gamma_{j,t}$$

where $[s/2]$ equals $s/2$ if s is even and equals $(s-1)/2$ if it is odd. The sinusoids $\gamma_{j,t}$ have frequencies $\lambda_j = 2\pi j/s$ and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma^*_{j,t} \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma^*_{j,t-1} \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega^*_{j,t} \end{bmatrix}$$

where the disturbances $\omega_{j,t}$ and $\omega^*_{j,t}$ are assumed to be independent and, for fixed j, $\omega_{j,t}$ and $\omega^*_{j,t} \sim N(0, \sigma_\omega^2)$. If s is even then the equation for $\gamma^*_{s/2,t}$ is not needed and $\gamma_{s/2,t}$ is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

Note that, whether the seasonal type is DUMMY or TRIGONOMETRIC, there is only one parameter, the disturbance variance σ_ω^2, in the seasonal model.

There can be more than one seasonal components in the model, necessarily with different season lengths. Each seasonal component is specified using a separate SEASONAL statement. A model with multiple seasonal components can easily become quite complex and may need large amount of data and computing resources for its estimation and forecasting. Currently, at most three seasonals can be included in a model. The following code examples illustrate the use of SEASON statement:

```
season length=4;
```

This statement specifies a DUMMY type (default), seasonal component with season length four, corresponding to the quarterly seasonality. The disturbance variance σ_ω^2 will be estimated from the data. The following statement specifies a trigonometric seasonal with monthly seasonality. It also provides a starting value for σ_ω^2.

```
      season length=12 type=trig variance=4;
```

LENGTH= *integer*

This option is used to specify the season length, *s*. This is a required option in this statement. The season length can be any integer larger than or equal to 2. Typical examples of season lengths are 12, corresponding to the monthly seasonality, or 4, corresponding to the quarterly seasonality.

NOEST

This option fixes the value of the the disturbance variance parameter to the value specified in the VARIANCE= option.

PRINT=FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)

This option requests printing of filtered or smoothed estimate of the seasonal component γ_t.

TYPE= DUMMY | TRIG

This option specifies the type of the seasonal component. The default type is DUMMY.

VARIANCE= *value*

This option is used to supply an initial value for the disturbance variance, σ_ω^2, in the γ_t equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

BLOCKSEASON Statement

BLOCKSEASON NBLOCKS= *integer*
BLOCKSIZE= *integer* < *options* > **;**

The BLOCKSEASON or BLOCKSEASONAL statement is used to specify a seasonal γ_t that has a special block structure. The seasonal γ_t is called a *block seasonal* of block size *m* and number of blocks *k* if its season length, *s*, can be factored as $s = m * k$ and its seasonal effects have a block form, that is, the first *m* seasonal effects are all equal to some number τ_1, the next *m* effects are all equal to some number τ_2, and so on. This type of seasonal structure can be appropriate in some cases, for example, consider a series that is recorded on an hourly basis. Further assume that, in this particular case, the *hour of the day* effect and the *day of the week* effect are *additive*. In this situation the hour of the week seasonality, having a season length of 168, can be modeled as a sum of two components. The hour of the day effect is modeled using a simple seasonal of season length 24, while the day of the week effect is modeled as a block seasonal that has the days of the week as blocks. This day of the week block seasonal will have seven blocks, each of size 24. A block seasonal specification requires, at the minimum, the block size *m* and the number of blocks in the seasonal *k*. These are specified using the BLOCKSIZE= and NBLOCKS= options, respectively. In addition, you may need to specify the position of the first observation of the series using the OFFSET= option, if it is not at the beginning of one of the blocks. In the example just considered, this will correspond to a situation where the first series measurement is not at the start of the day. Suppose that the first measurement of the series corresponds to the hour between 6:00 and 7:00 a.m., which is the seventh hour within that day or at the seventh position within that block. This is specified as OFFSET=7.

The other options of this statement are very similar to the options in the SEASONAL statement, for example, a block seasonal can also be of one of the two types, DUMMY or TRIGONOMETRIC. There can be more than one block seasonal component in the model, each specified using a separate BLOCKSEASON statement. No two block seasonals in the model can have the same NBLOCKS= and BLOCKSIZE= specifications. The following example illustrates the use of the BLOCKSEASON statement to specify the additive, hour of the week seasonal model:

```
season length=24 type=trig;
blockseason nblocks=7 blocksize=24;
```

BLOCKSIZE= *integer*

This option is used to specify the block size, *m*. This is a required option in this statement. The block size can be any integer larger than or equal to two. Typical examples of block sizes are 24, corresponding to the hours of the day when a day is being used as a block in hourly data, or 60, corresponding to the minutes in an hour when an hour is being used as a block in data recorded by minutes, etc.

NBLOCKS= *integer*

This option is used to specify the number of blocks, *k*. This is a required option in this statement. The number of blocks can be any integer larger than or equal to two.

NOEST

This option fixes the value of the the disturbance variance parameter to the value specified in the VARIANCE= option.

OFFSET= *integer*

This option is used to specify the position of the first measurement within the block, if the first measurement is not at the start of a block. The OFFSET= value must be between one and the block size. The default value is one. The *first measurement* refers to the start of the *estimation span* and the *forecast span*. If these spans differ, their starting measurements must be separated by an integer multiple of the block size.

PRINT=FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)

This option requests the printing of filtered or smoothed estimate of the block seasonal component γ_t.

TYPE= DUMMY | TRIG

This option specifies the type of the seasonal component. The default type is DUMMY.

VARIANCE= *value*

This option is used to supply an initial value for the disturbance variance, σ_ω^2, in the

γ_t equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

CYCLE Statement

CYCLE < *options* > **;**

The CYCLE statement is used to specify a *cycle* component, ψ_t, in the model. The stochastic equation governing a cycle component of period *p* and damping factor ρ is as follows:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where ν_t and ν_t^* are independent, zero mean, Gaussian disturbances with variance σ_ν^2 and $\lambda = 2 * \pi/p$ is the angular frequency of the cycle. Any *p* strictly larger than two is an admissible value for the period, and the damping factor ρ can be any value in the interval (0, 1), including one but excluding zero. The cycles with frequency zero and π, which correspond to the periods equal to infinity and two respectively, can be specified using the AUTOREG statement. The values of ρ smaller than one give rise to a stationary cycle, while $\rho = 1$ gives rise to a nonstationary cycle. As a default, values of ρ, *p*, and σ_ν^2 are estimated from the data. However, if necessary, you can fix the values of some, or all, of these parameters.

There can be multiple cycles in a model, each specified using a separate CYCLE statement. Currently, you can specify up to 50 cycles in a model.

The following examples illustrate the use of the CYCLE statement:

```
cycle;
cycle;
```

These statements request including two cycles in the model. The parameters of each of these cycles will be estimated from the data.

```
cycle rho=1 noest=rho;
```

This statement requests inclusion of a nonstationary cycle in the model. The cycle period *p* and the disturbance variance σ_ν^2 will be estimated from the data. In the following statement a nonstationary cycle with fixed period of 12 is specified. Moreover, a starting value is supplied for σ_ν^2.

```
cycle period=12 rho=1 variance=4 noest=(rho period);
```

NOEST=PERIOD
NOEST=RHO
NOEST=VARIANCE
NOEST= (< RHO > < PERIOD > < VARIANCE >)

This option fixes the values of the component parameters to those specified in RHO=, PERIOD=, and VARIANCE= options. This option enables you to fix any combination of parameter values.

PERIOD= *value*
This option is used to supply an initial value for the cycle period during the parameter estimation process. Period value must be strictly larger than 2.

PRINT=FILTER
PRINT= SMOOTH
PRINT= (FILTER SMOOTH)
This option requests the printing of a filtered or smoothed estimate of the cycle component ψ_t.

RHO= *value*
This option is used to supply an initial value for the damping factor in this component during the parameter estimation process. Any value in the interval (0, 1), including one but excluding zero, is an acceptable initial value for the damping factor.

VARIANCE= *value*
This option is used to supply an initial value for the disturbance variance parameter, σ_ν^2, to be used during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

AUTOREG Statement

AUTOREG < *options* > **;**

The AUTOREG statement specifies an *autoregressive* component of the model. An autoregressive component is a special case of cycle that corresponds to the frequency of zero or π. It is modeled separately for easier interpretation. A stochastic equation for an autoregressive component r_t can be written as follows:

$$r_t = \rho r_{t-1} + \nu_t, \quad \nu_t \sim i.i.d. \;\; N(0, \sigma_\nu^2)$$

The damping factor ρ can take any value in the interval (-1, 1), including -1 but excluding 1. If $\rho = 1$ the autoregressive component cannot be distinguished from the random walk level component. If $\rho = -1$ the autoregressive component corresponds to a seasonal component with season length 2, or a nonstationary cycle with period 2. If $|\rho| < 1$ then the autoregressive component is stationary. The following examples illustrate the AUTOREG statement:

```
autoreg;
```

This statement includes an autoregressive component in the model. The damping factor ρ and the disturbance variance σ_ν^2 are estimated from the data.

NOEST=RHO
NOEST= VARIANCE
NOEST= (RHO VARIANCE)

This option fixes the values of ρ and σ_ν^2 to those specified in RHO= and VARIANCE= options.

PRINT=FILTER
PRINT=SMOOTH
PRINT=(FILTER SMOOTH)
This option requests printing of filtered or smoothed estimate of the autoreg component.

RHO= *value*
This option is used to supply an initial value for the damping factor ρ during the parameter estimation process. The value of ρ must be in the interval (-1, 1), including -1 but excluding 1.

VARIANCE= *value*
This option is used to supply an initial value for the disturbance variance σ_ν^2 during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

DEPLAG Statement

DEPLAG LAGS= *order* **<PHI=** *value ...* **> < NOEST > ;**

The DEPLAG statement is used to specify the lags of the dependent variable to be included as predictors in the model. The following examples illustrate the use of DEPLAG statement:

```
deplag lags=2;
```

If the dependent series is denoted by y_t, this statement specifies the inclusion of $\phi_1 y_{t-1} + \phi_2 y_{t-2}$ in the model. The parameters ϕ_1 and ϕ_2 are estimated from the data. The following statement requests including $\phi_1 y_{t-1} + \phi_2 y_{t-4} - \phi_1 \phi_2 y_{t-5}$ in the model. The values of ϕ_1 and ϕ_2 are fixed at 0.8 and -1.2.

```
deplag lags=(1)(4) phi=0.8 -1.2 noest;
```

The dependent lag parameters are not constrained to lie in any particular region. In particular, this implies that a UCM that contains only an *irregular* component and dependent lags, resulting in a traditional autoregressive model, is not constrained to be a stationary model. In the DEPLAG statement if an initial value is supplied for any one of the parameters, the initial values must be supplied for all other parameters also.

LAGS= *order*
LAGS= (*lag, ..., lag***) ... (***lag, ..., lag***)**

This is a required option in this statement. LAGS=(l_1, l_2, ..., l_k) defines a model with specified lags of the dependent variable included as predictors. LAGS= *order* is equivalent to LAGS=(1, 2, ..., *order*).

A concatenation of parenthesized lists specifies a factored model. For example, LAGS=(1)(12) specifies that the lag values, 1, 12 and 13, corresponding to the following polynomial in the backward shift operator, be included in the model

$$(1 - \phi_{1,1}B)(1 - \phi_{2,1}B^{12})$$

Note that, in this case, the coefficient of the thirteenth lag is constrained to be the product of the coefficients of the first and twelfth lags.

PHI= *value* **...**
lists starting values for the coefficients of the lagged dependent variable.

NOEST
This option fixes the values of the parameters to those specified in PHI= options.

ESTIMATE Statement

ESTIMATE *< options >* **;**

The ESTIMATE statement is an optional statement used to control the overall model-fitting environment. Using this statement, you can control the span of observations used to fit the model using the SKIPFIRST= and BACK= options. This can be useful in model diagnostics. You can request a variety of goodness of fit statistics and other model diagnostic information. Note that this statement is not used to control the nonlinear optimization process itself. That is done using the NLOPTIONS statement where you can control the number of iterations, choose between the different optimization techniques, etc. The estimated parameters and other related information can be stored in a data set using the OUTEST= option. The following example illustrates the use of this statement:

```
estimate skipfirst=12 back=24;
```

This statement requests that the initial 12 measurements and the last 24 measurements be excluded during the model fitting process. The actual observation span used to fit the model is decided as follows: First the observations are scanned and the observation numbers of the first and last non-missing values of the dependent variable are determined. Suppose that the observation numbers of the first and the last non-missing values are n_0 and n_1, respectively. As a result of SKIPFIRST=12 and BACK=24, the measurements between observation numbers $n_0 + 11$ to $n_1 - 24$ form the estimation span. Of course, the model fitting may not take place if there are insufficient data in the resulting span. The model fitting will also not take place if there are regressors in the model that have missing values in the estimation span.

BACK= *integer*
SKIPLAST= *integer*
This option is used if some ending part of the data needs to be ignored during the parameter estimation. This can be useful when one wants to study the forecasting performance of the model on the observed data. SKIPLAST=10 results in skipping

the last 10 measurements of the response series during the parameter estimation. The default is SKIPLAST=0.

OUTEST= *SAS Dataset*
This option is used to specify an output data set for the estimated parameters.

PRINT=NONE
suppresses all the printed output related to the model fitting; for example, the parameter estimates, the goodness of fit statistics, etc.

SKIPFIRST= *integer*
This option is used if some early part of the data needs to be ignored during the parameter estimation. This can be useful if there is a reason to believe that the model being estimated is not appropriate for this portion of the data. SKIPFIRST=10 results in skipping the first 10 measurements of the response series during the parameter estimation. The default is SKIPFIRST=0.

FORECAST Statement

FORECAST < *options* > **;**

The FORECAST statement is an optional statement that is used to specify the overall forecasting environment for the specified model. It can be used to specify the span of observations, the historical period, to use to compute the forecasts of the future observations. This is done using the SKIPFIRST= and BACK= options. The number of periods to forecast beyond the historical period, and the significance level of the forecast confidence interval, are specified using the LEAD= and ALPHA= options. You can request one step ahead series and component forecasts using the PRINT= option. The series forecasts, and the model based decomposition of the series, can be saved in a data set using the OUTFOR= option. The following example illustrates the use of this statement:

```
forecast skipfirst=12 back=24 lead=30;
```

This statement requests that the initial 12 measurements and the last 24 response values be excluded during the forecast computations. The forecast horizon is 30 periods, that is multi step forecasting will begin at the end of the historical period and continue for 30 periods. The actual observation span used to compute the multi step forecasting is decided as follows: First the observations are scanned and the observation numbers of the first and last non-missing values of the response variable are determined. Suppose that the observation numbers of the first and last non-missing values are n_0 and n_1, respectively. As a result of SKIPFIRST=12 and BACK=24, the historical period, or the forecast span, begins at $n_0 + 12$ and ends at $n_1 - 24$. Multi step forecasts are produced for the next 30 periods, that is, for the observation numbers $n_1 - 23$ to $n_1 + 6$. Of course, the forecast computations may fail because of insufficient data in the forecast span. It can also fail if the model has regressor variables that have missing values in the forecast span. If the regressors contain missing values in the forecast horizon, that is between the observations $n_1 - 23$ to $n_1 + 6$, the forecast horizon is reduced accordingly.

ALPHA= *value*
specifies the significance level of the forecast confidence intervals, e.g., ALPHA=0.05 results in a 95% confidence interval.

BACK= *integer*
SKIPLAST= *integer*
This can be useful to specify the holdout sample for the evaluation of the forecasting performance of the model. SKIPLAST=10 results in treating the last 10 observed values of the response series as being unobserved. The default is SKIPLAST=0.

LEAD= *integer*
This option is used to specify the number of periods to forecast beyond the historical period defined by the SKIPFIRST= and SKIPLAST= options, for example, LEAD=10 will result in the forecasting of 10 future values of the response series. The default is LEAD=12.

OUTFOR= *SAS Dataset*
This option is used to specify an output data set for the forecasts. The output data set contains the ID variable (if specified), the response and predictor series, the one step ahead and out of sample response series forecasts, the forecast confidence intervals, the smoothed values of the response series and, the smoothed forecasts produced as a result of the model-based decomposition of the series.

PRINT=DECOMP
PRINT=FORECASTS
PRINT=NONE
PRINT=(FORECASTS DECOMP)
This option can be used to control the printing of the series forecasts and the printing of smoothed model decomposition estimates. By default, the series forecasts are printed only for the forecast horizon specified by the LEAD= option, that is, the one step ahead forecasts during the entire forecast span are not printed. You can request forecasts for the entire forecast span by specifying the PRINT=FORECASTS option. Using the PRINT=DECOMP, you can get smoothed estimates of the following effects: trend, trend plus regression, trend plus regression plus cycle, and sum of all components except the irregular. If some of these effects are absent in the model then they are ignored. You can use PRINT=NONE to suppress the printing of all of the forecast output.

SKIPFIRST= *integer*
This option is used if some early part of the data needs to be ignored during the forecasting calculations. This can be useful if there is a reason to believe that the model being used for forecasting is not appropriate for this portion of the data. SKIPFIRST=10 results in skipping the first 10 measurements of the response series during the forecast calculations. The default is SKIPFIRST=0.

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

PROC UCM uses the NonLinear Optimization (NLO) subsystem to perform the nonlinear optimization of the likelihood function during the estimation of model parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. For most problems the default settings of the optimization process are adequate, however, in some cases it may be useful to change the optimization technique or to change the maximum number of iterations. This can be done by using the TECH= and MAXITER= options in the NLOPTIONS statement as follows

```
nloptions tech=dbldog maxiter=200;
```

This will set the maximum number of iterations to 200 and change the optimization technique to DBLDOG rather than the default technique, TRUREG, used in PROC UCM. A discussion of the full range of options that can be used with the NLOPTIONS statement is given in the chapter on Nonlinear Optimization Methods (Chapter 10, "Nonlinear Optimization Methods."). In PROC UCM all these options are available except the options related to the printing of the optimization history. In this version of PROC UCM all the printed output from the NLO subsystem is suppressed.

Details

Throughout this section, $Diag\,[a, b, \ldots\,]$ will denote a diagonal matrix with diagonal entries $[a, b, \ldots\,]$, and the transpose of a matrix T will be denoted as $T^{'}$.

The UCMs as State Space Models

It is well known that the UCMs considered in PROC UCM can be thought of as special cases of more general models, the (linear) Gaussian State Space Models (GSSM). A GSSM suitable for our purposes can be described as follows:

$$\begin{aligned} y_t &= Z_t \alpha_t \\ \alpha_{t+1} &= T_t \alpha_t + \zeta_{t+1}, \qquad \zeta_t \sim \mathrm{N}(0, Q_t) \\ \alpha_1 &\sim \mathrm{N}(0, P) \end{aligned}$$

The first equation, called the *Observation Equation*, relates the observed series y_t to a state vector α_t that is usually unobserved. The second equation describes the evolution of the state vector in time and is called the *State Equation*. The system matrices Z_t and T_t are of appropriate dimensions and are known, except possibly for some unknown elements that become part of the parameter vector of the model. The noise series ζ_t consists of independent, zero mean, Gaussian vectors with covariance matrices Q_t. For most of the UCMs considered here, the system matrices Z_t and T_t, and the noise covariances Q_t, are time invariant, i.e., they do not depend on time. In a few cases, however, some or all of them may depend on time. The initial state vector

α_1 is assumed to be independent of the noise series, and its covariance matrix P can be partially *diffuse*. A random vector has a partially diffuse covariance matrix if it can be partitioned such that one part of the vector has a properly defined probability distribution, while the covariance matrix of the other part is infinite, i.e., you have no prior information about this part of the vector. The covariance of the initial state α_1 is assumed to have the following form:

$$P = P_* + \kappa P_\infty$$

where P_* and P_∞ are nonnegative definite, symmetric matrices and κ is a constant that is assumed to be close to ∞. In the case of UCMs considered here P_∞ is always a diagonal matrix consisting of zeros and ones, and, if a particular diagonal element of P_∞ is one, then the corresponding row and column in P_* is zero.

The state space formulation of a UCM has many computational advantages. In this formulation there are convenient algorithms for estimating and forecasting the unobserved states $\{\alpha_t\}$ using the observed series $\{y_t\}$. These algorithms also yield the in-sample and out of sample forecasts and the likelihood of $\{y_t\}$. The state space representation of a UCM need not be unique. In the representation used here, the unobserved components in the UCM are taken as elements of the state vector. This makes the elements of the state interpretable and, more importantly, the sample estimates and forecasts of these unobserved components are easily obtained. For additional information on the computational aspects of the state space modeling, see Durbin and Koopman (2001). Next some notation is developed to describe the essential quantities computed during the analysis of the state space models.

Let $\{y_t, t = 1, \ldots, n\}$ be the observed sample from a series satisfying a state space model. Next, for $1 \le t \le n$, let the one step ahead forecasts of the series and the states, and their variances, be defined as follows:

$$\begin{aligned}
\hat{\alpha}_t &= E(\alpha_t | y_1, y_2, \ldots, y_{t-1}) \\
\Gamma_t &= Var(\alpha_t | y_1, y_2, \ldots, y_{t-1}) \\
\hat{y}_t &= E(y_t | y_1, y_2, \ldots, y_{t-1}) \\
F_t &= Var(y_t | y_1, y_2, \ldots, y_{t-1})
\end{aligned}$$

using the usual notation to denote the conditional expectations and conditional variances. These are also called the filtered estimates of the series and the states. Similarly, for $t \ge 1$, let

$$\begin{aligned}
\tilde{\alpha}_t &= E(\alpha_t | y_1, y_2, \ldots, y_n) \\
\Delta_t &= Var(\alpha_t | y_1, y_2, \ldots, y_n) \\
\tilde{y}_t &= E(y_t | y_1, y_2, \ldots, y_n) \\
G_t &= Var(y_t | y_1, y_2, \ldots, y_n)
\end{aligned}$$

denote the full-sample estimates of the series and the state values at time t. If the time t is in the historical period, i.e., if $1 \le t \le n$, then the full-sample estimates are called the *smoothed* estimates, and if t lies in the future then they are called out of sample forecasts. Note that, if $1 \le t \le n$, $\tilde{y}_t = y_t$ and $G_t = 0$, unless y_t is missing.

All the filtered and smoothed estimates $\hat{\alpha}_t, \tilde{\alpha}_t, \ldots, G_t$ are computed using the filtering and smoothing algorithms given in Durbin and Koopman (2001). These algorithms are iterative. If the initial state is diffuse, the effect of the improper prior distribution of α_1 manifests itself in the first few filtering iterations. During these initial filtering iterations the distribution of the filtered quantities remains diffuse, that is, during these iterations the one step ahead series and state forecast variances F_t and Γ_t have the following form

$$\begin{aligned}\Gamma_t &= \Gamma_{*t} + \kappa \Gamma_{\infty t}\\ F_t &= F_{*t} + \kappa F_{\infty t}\end{aligned}$$

The actual number of iterations, say d, affected by this improper prior depends on the nature of the matrix sequence T_t, the rank of P_∞, and the pattern of missing values in the dependent series. After d iterations, $\Gamma_{\infty t}$ and $F_{\infty t}$ become zero and the one step ahead series and state forecasts have proper distributions. In certain missing value patterns it can happen that d exceeds the sample size; that is, the sample information is insufficient to create a proper prior for the filtering process. In these cases no parameter estimation or forecasting is done. The forecasting computations can also fail if the specified model contains components that are essentially multicollinear and the process of computing one step ahead or multi step ahead forecasts is unstable. This condition can also manifest itself as failure to initialize a proper prior for the filtering process.

The log-likelihood of the sample, which takes account of this diffuse initialization steps, is computed using the one step ahead series forecasts as follows

$$\log L_d(y_1, \ldots, y_n) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{d} w_t - \frac{1}{2}\sum_{t=d+1}^{n}(\log F_t + \frac{\nu_t^2}{F_t})$$

where $\nu_t = y_t - Z_t\hat{\alpha}_t$ are the one step ahead residuals and

$$\begin{aligned}w_t &= \log F_{\infty t} && \text{if } F_{\infty t} > 0\\ &= \log F_{*t} + \frac{\nu_t^2}{F_{*t}} && \text{if } F_{\infty t} = 0\end{aligned}$$

If y_t is missing at some time t, then the corresponding summand in the log-likelihood expression is deleted, and the constant term is adjusted suitably.

The portion of the log-likelihood corresponding to the post-initialization period is called the non-diffuse log-likelihood. The non-diffuse log-likelihood is given by

$$\log L(y_1, \ldots, y_n) = -\frac{1}{2}\sum_{t=d+1}^{n}(\log F_t + \frac{\nu_t^2}{F_t})$$

In the case of UCMs considered in PROC UCM, it often happens that the diffuse part of the likelihood, $\sum_{t=1}^{d} w_t$, does not depend on the model parameters, and in these cases the maximization of non-diffuse and diffuse likelihoods is equivalent. However, in some cases, for example, when the model consists of dependent lags, the diffuse part does depend on the model parameters. In these cases the maximization of the diffuse and non-diffuse likelihood can produce different results.

In the remainder of this section the state space formulation of UCMs is further explained using some particular UCMs as examples. The examples will show that the state space formulation of the UCMs depends upon the components in the model in a simple fashion; for example, the system matrix T will usually be a block diagonal matrix with blocks corresponding to the components in the model. The only exception to this pattern is the UCMs consisting of the lags of dependent variable. This case is considered at the end of the section.

Local Level Model

Recall that the dynamics of a local level model are

$$\begin{aligned} y_t &= \mu_t + \epsilon_t \\ \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \\ \beta_t &= \beta_{t-1} + \xi_t \end{aligned}$$

Here y_t is the response series and ϵ_t, η_t, and ξ_t are independent, mean zero Gaussian disturbance sequences with variances $\sigma^2_\epsilon, \sigma^2_\eta$ and σ^2_ξ respectively. This model can be formulated as a state space model where the state vector $\alpha_t = [\ \epsilon_t\ \mu_t\ \beta_t\ ]'$ and the state noise $\zeta_t = [\ \epsilon_t\ \eta_t\ \xi_t\ ]'$. Note that the elements of the state vector are precisely the unobserved components in the model. The system matrices T, Z and the noise covariance Q corresponding to this choice of state and state noise vectors can be seen to be time invariant and are given by

$$Z = [\ 1\ 1\ 0\ ], \quad T = \begin{bmatrix} 0\ 0\ 0 \\ 0\ 1\ 1 \\ 0\ 0\ 1 \end{bmatrix} \quad \text{and} \quad Q = Diag\left[\sigma^2_\epsilon, \sigma^2_\eta, \sigma^2_\xi\right]$$

The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma^2_\epsilon, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1]$. The parameter vector θ consists of all the disturbance variances, that is, $\theta = (\sigma^2_\epsilon, \sigma^2_\eta, \sigma^2_\xi)$.

Basic Structural Model

Basic Structural Model (BSM) is obtained by adding a seasonal component, γ_t, to the local level model. In order to economize on the space, the state space formulation of a BSM with relatively short season length, season length = 4 (quarterly seasonality), is considered here. The pattern for longer season lengths such as 12 (monthly) and 52 (weekly) is easy to see.

Let us first consider the dummy form of seasonality. In this case the state and the state noise vectors are $\alpha_t = [\ \epsilon_t\ \mu_t\ \beta_t\ \gamma_{1,t}\ \gamma_{2,t}\ \gamma_{3,t}\ ]'$ and $\zeta_t = [\ \epsilon_t\ \eta_t\ \xi_t\ \omega_t\ 0\ 0\ ]'$. The first three elements of the state vector are the irregular, level, and the slope components, respectively. The remaining elements, $\gamma_{i,t}$, are lagged versions of the seasonal component γ_t. $\gamma_{1,t}$ corresponds to lag zero, that is, the same as γ_t, $\gamma_{2,t}$ to lag 1 and $\gamma_{3,t}$ to lag 2. The system matrices can be seen to be

$$Z = [\,1\;1\;0\;1\;0\;0\,],\quad T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{-1} & \text{-1} & \text{-1} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and $Q = Diag\left[\sigma^2_\epsilon, \sigma^2_\eta, \sigma^2_\xi, \sigma^2_\omega, 0, 0\right]$. The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma^2_\epsilon, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1, 1, 1, 1]$.

In the case of trigonometric form of seasonality, $\alpha_t = \left[\;\epsilon_t\;\mu_t\;\beta_t\;\gamma_{1,t}\;\gamma^*_{1,t}\;\gamma_{2,t}\;\right]'$ and $\zeta_t = \left[\;\epsilon_t\;\eta_t\;\xi_t\;\omega_{1,t}\;\omega^*_{1,t}\;\omega_{2,t}\;\right]'$. The disturbance sequences, $\omega_{j,t}, 1 \leq j \leq 2$, and $\omega^*_{1,t}$ are independent, zero mean, Gaussian sequences with variance σ^2_ω.

$$Z = [\,1\;1\;0\;1\;0\;1\,],\quad T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\lambda_1 & \sin\lambda_1 & 0 \\ 0 & 0 & 0 & -\sin\lambda_1 & \cos\lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\lambda_2 \end{bmatrix}$$

and $Q = Diag\left[\sigma^2_\epsilon, \sigma^2_\eta, \sigma^2_\xi, \sigma^2_\omega, \sigma^2_\omega, \sigma^2_\omega\right]$. Here $\lambda_j = (2\pi j)/4$. The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma^2_\epsilon, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1, 1, 1, 1]$. The parameter vector, in both the cases, is $\theta = (\sigma^2_\epsilon, \sigma^2_\eta, \sigma^2_\xi, \sigma^2_\omega)$.

Seasonals with Blocked Seasonal Values

Block seasonals are special seasonals that impose a special block structure on the seasonal effects. Let us consider a BSM with monthly seasonality that has a quarterly block structure, that is, months within the same quarter are assumed to have identical effects except for some random perturbation. Such a seasonal is a block seasonal with block size m equal to 3 and the number of blocks k equal to 4. The state space structure for such a model with DUMMY type seasonality is as follows: The state and the state noise vectors are $\alpha_t = \left[\;\epsilon_t\;\mu_t\;\beta_t\;\gamma_{1,t}\;\gamma_{2,t}\;\gamma_{3,t}\;\right]'$ and $\zeta_t = \left[\;\epsilon_t\;\eta_t\;\xi_t\;\omega_t\;0\;0\;\right]'$. The first three elements of the state vector are the irregular, level, and the slope components, respectively. The remaining elements, $\gamma_{i,t}$, are lagged versions of the seasonal component γ_t. $\gamma_{1,t}$ corresponds to lag zero, that is, the same as γ_t, $\gamma_{2,t}$ to lag

m and $\gamma_{3,t}$ to lag $2m$. All the system matrices are time invariant, except the matrix T. They can be seen to be $Z = [\ 1\ 1\ 0\ 1\ 0\ 0\ ]$, $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, 0, 0\right]$, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \text{-1} & \text{-1} & \text{-1} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

when t is a multiple of the block size m, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

otherwise. Note that when t is not a multiple of m, the portion of the T_t matrix corresponding to the seasonal is identity. The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1, 1, 1, 1]$.

Similarly in the case of trigonometric form of seasonality, $\alpha_t = \left[\ \epsilon_t\ \mu_t\ \beta_t\ \gamma_{1,t}\ \gamma_{1,t}^*\ \gamma_{2,t}\ \right]'$ and $\zeta_t = \left[\ \epsilon_t\ \eta_t\ \xi_t\ \omega_{1,t}\ \omega_{1,t}^*\ \omega_{2,t}\ \right]'$. The disturbance sequences, $\omega_{j,t}, 1 \le j \le 2$, and $\omega_{1,t}^*$ are independent, zero mean, Gaussian sequences with variance σ_ω^2. $Z = [\ 1\ 1\ 0\ 1\ 0\ 1\ ]$, $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, \sigma_\omega^2, \sigma_\omega^2\right]$, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\lambda_1 & \sin\lambda_1 & 0 \\ 0 & 0 & 0 & -\sin\lambda_1 & \cos\lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\lambda_2 \end{bmatrix}$$

when t is a multiple of the block size m, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

otherwise. As before, when t is not a multiple of m, the portion of the T_t matrix corresponding to the seasonal is identity. Here $\lambda_j = (2\pi j)/4$. The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\left[0, 1, 1, 1, 1, 1\right]$. The parameter vector, in both the cases, is $\theta = (\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2)$.

Cycles and Auto-Regression

The preceding examples have illustrated how to build a state space model corresponding to a UCM that includes components such as irregular, trend, and seasonal. There one can see that the state vector and the system matrices have a simple block structure with blocks corresponding to the components in the model. Therefore, here only a simple model consisting of a single cycle and an irregular component is considered. The state space form for more complex UCMs consisting of multiple cycles and other components can be easily deduced from this example.

Recall that a stochastic cycle ψ_t with frequency λ, $0 < \lambda < \pi$, and damping coefficient ρ can be modeled as

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where ν_t and ν_t^* are independent, zero mean Gaussian disturbances with variance σ_ν^2. In what follows, a state space form for a model consisting of such a stochastic cycle and an irregular component is given.

The state vector $\alpha_t = \left[\, \epsilon_t \; \psi_t \; \psi_t^* \,\right]'$, state noise vector $\zeta_t = \left[\, \epsilon_t \; \nu_t \; \nu_t^* \,\right]'$. The system matrices are

$$Z = [\,1\;1\;0\,] \quad T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \rho\cos\lambda & \rho\sin\lambda \\ 0 & -\rho\sin\lambda & \rho\cos\lambda \end{bmatrix} \quad Q = Diag\left[\sigma_\epsilon^2, \sigma_\nu^2, \sigma_\nu^2\right]$$

The distribution of the initial state vector α_1 is proper with $P_* = Diag\left[\sigma_\epsilon^2, \sigma_\psi^2, \sigma_\psi^2\right]$ where $\sigma_\psi^2 = \sigma_\nu^2(1-\rho^2)^{-1}$. The parameter vector $\theta = (\sigma_\epsilon^2, \rho, \lambda, \sigma_\nu^2)$.

An auto-regression r_t can be considered as a special case of cycle with frequency λ equal to 0 or π. In this case the equation for ψ_t^* is not needed. Therefore, for a UCM consisting of an auto-regressive component and an irregular component, the state space model simplifies to the following form:

The state vector $\alpha_t = \left[\, \epsilon_t \; r_t \,\right]'$, state noise vector $\zeta_t = \left[\, \epsilon_t \; \nu_t \,\right]'$. The system matrices are

$$Z = [\,1\;1\,], \quad T = \begin{bmatrix} 0 & 0 \\ 0 & \rho \end{bmatrix} \quad \text{and} \quad Q = Diag\left[\sigma_\epsilon^2, \sigma_\nu^2\right]$$

The distribution of the initial state vector α_1 is proper with $P_* = Diag\left[\sigma_\epsilon^2, \sigma_r^2\right]$ where $\sigma_r^2 = \sigma_\nu^2(1-\rho^2)^{-1}$. The parameter vector $\theta = (\sigma_\epsilon^2, \rho, \sigma_\nu^2)$.

Incorporating the Predictors

As with earlier examples, how to obtain a state space form of a UCM consisting of predictors is illustrated using a simple special case. Consider a random walk model with predictors x_1 and x_2. The dynamics of this model are

$$\begin{aligned} y_t &= \mu_t + \beta_1 x_{1t} + \beta_2 x_{2t} + \epsilon_t \\ \mu_t &= \mu_{t-1} + \eta_t \end{aligned}$$

This dynamics can be captured in the state space form by taking $\alpha_t = [\ \epsilon_t\ \mu_t\ \beta_1\ \beta_2\ ]^{'}$, $\zeta_t = [\ \epsilon_t\ \eta_t\ 0\ 0\ ]^{'}$, and

$$Z_t = [\ 1\ 1\ x_{1t}\ x_{2t}\ ] \quad T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q = Diag\left[\sigma^2_\epsilon, \sigma^2_\eta, 0, 0\right]$$

Note that the regression coefficients are elements of the state vector, and that the system matrix Z_t is not time invariant. The distribution of the initial state vector α_1 is diffuse with $P_* = Diag\left[\sigma^2_\epsilon, 0, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1, 1]$. The parameters of this model are the disturbance variances, σ^2_ϵ and σ^2_η, and the regression coefficients, β_1 and β_2. The disturbance variances, being elements of the system matrix Q, are estimated by maximizing the likelihood, while the regression parameters get implicitly estimated during the state estimation (smoothing).

Models with Dependent Lags

The state space form of a UCM consisting of the lags of the dependent variable is quite different from the state space forms considered so far. Let us consider an example to illustrate this situation. Suppose that the preceding random walk with predictors model also includes a few, say k, lags of the dependent variable. That is

$$\begin{aligned} y_t &= \sum_{i=1}^{k} \phi_i y_{t-i} + \mu_t + \beta_1 x_{1t} + \beta_2 x_{2t} + \epsilon_t \\ \mu_t &= \mu_{t-1} + \eta_t \end{aligned}$$

The state space form of this augmented model can be described in terms of the state space form of the preceding random walk with predictors model. A superscript † has been added to distinguish the augmented model state space entities from the corresponding entities of the state space form of the random walk with predictors model. With this notation, the state vector of the augmented model $\alpha^\dagger_t = \left[\ \alpha^{'}_t\ y_t\ y_{t-1}\ \ldots\ y_{t-k+1}\ \right]^{'}$ and the new state noise vector $\zeta^\dagger_t = \left[\ \zeta^{'}_t\ u_t\ 0 \ldots\ 0\ \right]^{'}$ where u_t is the matrix product $Z_t\zeta_t$. Note that the length of the new state vector is $k + \text{length}(\alpha_t) = k + 4$. The new system matrices, in the block form, are

$$Z^\dagger_t = [\ 0\ 0\ 0\ 0\ 1\ \ldots\ 0\ ], \quad T^\dagger_t = \begin{bmatrix} T_t & 0 & \ldots & 0 \\ Z_{t+1}T_t & \phi_1 & \ldots & \phi_k \\ 0 & I_{k-1,k-1} & & 0 \end{bmatrix}$$

where $I_{k-1,k-1}$ is the $k-1$ dimensional identity matrix and,

$$Q_t^\dagger = \begin{bmatrix} Q_t & Q_t Z_t' & 0 \\ Z_t Q_t & Z_t Q_t Z_t' & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Note that the T and Q matrices of the random walk with predictors model are time invariant, in the expressions above their time indices are kept because it illustrates the pattern for more general models. The initial state vector is diffuse with

$$P_*^\dagger = \begin{bmatrix} P_* & 0 \\ 0 & 0 \end{bmatrix} \quad P_\infty^\dagger = \begin{bmatrix} P_\infty & 0 \\ 0 & I_{k,k} \end{bmatrix}$$

The parameters of this model are the disturbance variances σ_ϵ^2 and σ_η^2, the lag coefficients $\phi_1, \phi_2, \ldots, \phi_k$, and the regression coefficients β_1 and β_2. As before, the regression coefficients get estimated during the state smoothing, and the other parameters are estimated by maximizing the likelihood.

Missing Values

Embedded missing values in the dependent variable usually cause no problems in UCM modeling; however, no embedded missing values are allowed in the predictor variables. Certain patterns of missing values in the dependent variable can create problems for some models. For example, if in a monthly series all values are missing for a certain month, say May, then a BSM with monthly seasonality cannot be fit to these data. A non-seasonal model such as a local linear model can still be fit to these data.

Parameter Estimation

The parameter vector in a UCM consists of the variances of the disturbance terms of the unobserved components, the damping coefficients and frequencies in the cycles, the damping coefficient in the autoregression, the lag coefficients of the dependent lags, and the regression coefficients in the regression terms. The regression coefficients are always part of the state vector and are estimated by state smoothing. The remaining parameters are estimated by maximizing either the full diffuse likelihood or the non-diffuse likelihood. The decision to use the full diffuse likelihood or the non-diffuse likelihood depends on the presence or absence of the dependent lag coefficients in the parameter vector. If the parameter vector does not contain any dependent lag coefficients, then the full diffuse likelihood is used. If, on the other hand, the parameter vector does contain some dependent lag coefficients, then the parameters are estimated by maximizing the non-diffuse likelihood. The optimization of the full diffuse likelihood is often unstable when the parameter vector contains dependent lag coefficients. In this sense, when the parameter vector contains dependent lag coefficients the parameter estimates are not true maximum likelihood estimates.

The optimization of the likelihood, either the full or the non-diffuse, is carried out using one of several non-linear optimization algorithms. The user can control many aspects of the optimization process using the NLOPTIONS statement and by providing the starting values of the parameters while specifying the corresponding components. However, in most cases the default settings work quite well. The optimization process is not guaranteed to converge to a maximum likelihood estimate. In most cases the difficulties in parameter estimation are associated with the specification of a model that is not appropriate for the series being modeled.

t-values

The *t* values reported in the table of parameter estimates are approximations whose accuracy depends on the validity of the model, the nature of the model, and the length of the observed series. The distributional properties of the maximum likelihood estimates of general unobserved components models have not been explored fully, therefore the probability values corresponding to a *t* distribution should be interpreted carefully as they may be misleading. This is particularly true if the parameters in question are close to the boundary of the parameter space. Harvey (1989, 2001) are good references for information on this topic.

Computer Resource Requirements

The computing resources required for the UCM procedure depend on several factors. The memory requirement for the procedure is largely dependent on the number of observations to be processed and the size of the state vector underlying the specified model. If n denotes the sample size and m denotes the size of the state vector, the memory requirement is of the order of $6 \times 8 \times n \times m^2$ bytes, ignoring the lower order terms. The computing time for the parameter estimation also depends on m and n, as well as on the type of components included in the model. For example, the parameter estimation is usually faster if the model parameter vector consists only of disturbance variances, because in this case there is an efficient way to compute the likelihood gradient.

Printed Output

The default printed output produced by the UCM procedure is described in the following list:

- brief information about the input data set that includes the dataset name and label, and the name of the ID variable specified in the ID statement
- summary statistics for the data in the estimation and forecast spans, which includes the names of the variables in the model, their categorization as the dependent or predictor, the index of the beginning and the ending observations in the spans, the total number of observations and the number of missing observations, the smallest and the largest measurements, and the mean and the standard deviation
- information about the model parameters at the start of the model fitting stage that includes the fixed parameters in the model and the initial estimates of the free parameters in the model

- convergence status of the likelihood optimization process if any parameter estimation is done
- estimates of the free parameters at the end of the model fitting stage that includes the parameter estimates, their approximate standard errors, *t*-statistics, and the approximate P-value
- the likelihood-based goodness of fit statistics that include the full likelihood, the portion of the likelihood corresponding to the diffuse initialization, the sum of squares of the *innovations* normalized by their standard errors, and the Akike and the Bayesian information criteria AIC and BIC
- the fit statistics that are based on the raw residuals (observed - predicted), that include the mean squared error (MSE), root mean squared error (RMSE), the mean absolute percentage error (MAPE), the maximum percentage error (MAXPE), the R-Square, the adjusted R-Square, the Random Walk R-Square, and the Amemiya's R-Square
- the significance analysis of the components included in the model that is based on the estimation span
- brief information about the cycles, seasonals, and block seasonals present in the model
- the multi step series forecasts

ODS Table Names

The UCM procedure assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table:

Table 29.1. ODS Tables Produced in the UCM Procedure

ODS Table Name	Description	Option
ODS Tables Summarizing the Estimation and Forecast Spans		
EstimationSpan	Estimation span summary information	default
ForecastSpan	Forecast span summary information	default
ODS Tables Related to the Model Parameters		
ConvergenceStatus	Convergence status of the estimation process	default
FixedParameters	Fixed parameters in the model	default
InitialParameters	Initial estimates of the free parameters	default
ParameterEstimates	Final estimates of the free parameters	default
ODS Tables Related to the Model Information and Diagnostics		
BlockSeasonDescription	Information about the block seasonals in the model	default

Table 29.1. (ODS Tables Continued)

ODS Table Name	Description	Option
ComponentSignificance	Significance analysis of the components in the model	default
CycleDescription	Information about the cycles in the model	default
FitStatistics	Fit statistics based on the one step ahead predictions	default
FitSummary	Likelihood based fit statistics	default
SeasonDescription	Information about the seasonals in the model	default
ODS Tables Related to the Component Estimates		
FilteredAutoReg	Filtered Estimate of an Autoreg Component. AUTOREG statement	PRINT=FILTER
FilteredBlockSeason	Filtered Estimate of a Block Seasonal Component. BLOCKSEASON statement	PRINT=FILTER
FilteredCycle	Filtered Estimate of a Cycle Component. CYCLE statement	PRINT=FILTER
FilteredIrregular	Filtered Estimate of the Irregular Component. IRREGULAR statement	PRINT=FILTER
FilteredLevel	Filtered Estimate of the Level Component. LEVEL statement	PRINT=FILTER
FilteredSeason	Filtered Estimate of a Seasonal Component. SEASON statement	PRINT=FILTER
FilteredSlope	Filtered Estimate of the Slope Component. SLOPE statement	PRINT=FILTER
SmoothedAutoReg	Smoothed Estimate of an Autoreg Component. AUTOREG statement	PRINT=SMOOTH
SmoothedBlockSeason	Smoothed Estimate of a Block Seasonal Component. BLOCKSEASON statement	PRINT=SMOOTH
SmoothedCycle	Smoothed Estimate of the Cycle Component. CYCLE statement	PRINT=SMOOTH
SmoothedIrregular	Smoothed Estimate of the Irregular Component. IRREGULAR statement	PRINT=SMOOTH
SmoothedLevel	Smoothed Estimate of the Level Component. LEVEL statement	PRINT=SMOOTH
SmoothedSeason	Smoothed Estimate of a Seasonal Component. SEASON statement	PRINT=SMOOTH
SmoothedSlope	Smoothed Estimate of the Slope Component. SLOPE statement	PRINT=SMOOTH
ODS Tables Related to the FORECAST Statement		

Table 29.1. (ODS Tables Continued)

ODS Table Name	**Description**	**Option**
Forecasts	Dependent Series Forecasts	default
SmoothedAllExceptIrreg	Smoothed estimate of sum of all components except the irregular component	PRINT=DECOMP
SmoothedTrend	Smoothed estimate of trend	PRINT= DECOMP
SmoothedTrendReg	Smoothed estimate of trend plus regression	PRINT=DECOMP
SmoothedTrendRegCyc	Smoothed estimate of trend plus regression plus cycles and autoreg	PRINT=DECOMP

NOTE: The tables are related to a single series within a BY group. In the case of models that contain multiple cycles, seasonals, or block seasonals, the corresponding component estimate tables are sequentially numbered. For example, if a model contains two cycles and a seasonal and PRINT=SMOOTH option is used for each of them, the ODS tables containing the smoothed estimates will be named as SmoothedCycle1, SmoothedCycle2, and SmoothedSeason. Note that the seasonal table is not numbered because there is only one seasonal.

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the UCM procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, “Statistical Graphics Using ODS.”

When the ODS GRAPHICS are in effect, the UCM procedure produces a variety of plots. The main types of plots available are as follows:

- Time series plots of the component estimates, either filtered or smoothed, can be requested using the PLOT= option in the respective component statements. For example, the use of PLOT=SMOOTH option in a CYCLE statement produces a plot of smoothed estimate of that cycle.
- Residual plots for model diagnostics can be obtained using the PLOT= option in the ESTIMATE statement.
- Plots of series forecasts and model decompositions can be obtained using the PLOT= option in the FORECAST statement.

The details of the PLOT= option in different statements are given below.

The PLOT= option in the IRREGULAR, LEVEL, SLOPE, CYCLE, and AUTOREG statements is identical. You can use the FILTER and SMOOTH options to plot the filtered and smoothed estimates of the respective components as follows:

PLOT= FILTER
PLOT= SMOOTH
PLOT= (FILTER SMOOTH)

In the SEASON and BLOCKSEASON statements, the PLOT= option is as follows:

PLOT= FILTER
PLOT= SMOOTH
PLOT= F_ANNUAL
PLOT= S_ANNUAL
PLOT= (<FILTER> <SMOOTH> <F_ANNUAL> <S_ANNUAL>)

You can use the FILTER and SMOOTH options to plot the filtered and smoothed estimates of the seasonal or block seasonal component γ_t. The F_ANNUAL and S_ANNUAL options can be used to get the plots of "annual" variation in the filtered and smoothed estimates of γ_t. The annual plots are useful to see the change in the contribution of a particular month over the span of years. Here the "month" and the "year" are generic terms that change appropriately with the interval type being used to label the observations and the season length. For example, for monthly data with a season length of 12, the usual meaning applies, while for daily data with a season length of 7, the days of the week serve as months and the weeks serve as years.

In the ESTIMATE statement, the PLOT= option is used to obtain different residual diagnostic plots. The different possibilities are as follows:

PLOT= RESIDUAL
PLOT= ACF
PLOT= NORMAL
PLOT= WN
PLOT= (<RESIDUAL> <ACF> <NORMAL> <WN>)

The RESIDUAL option results in the residual plot, the ACF option gives the plot of the auto-correlations of the residuals, the NORMAL option gives the histogram of residuals, and the WN option gives the plot of White Noise test probabilities.

In the FORECAST statement, the PLOT= option can be used to obtain forecast and model decomposition plots. The details are as follows:

PLOT= FORECASTS
PLOT= TREND
PLOT= DECOMP
PLOT= DECOMPVAR
PLOT= FDECOMP
PLOT= FDECOMPVAR
PLOT= (<FORECASTS> <TREND> <DECOMP> <DECOMPVAR> <FDECOMP>

<FDECOMPVAR>)

The FORECASTS option provides the plot of the series forecasts, the TREND and DECOMP options provide the plots of the smoothed trend and other decompositions, the DECOMPVAR option can be used to plot the variance of these components, and the FDECOMP and FDECOMPVAR provide the same plots for the filtered decomposition estimates and their variances.

For an example of how to set up the ODS GRAPHICS environment and use the different PLOT= options, see Example 29.4.

ODS Graph Names

PROC UCM assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 29.2.

Table 29.2. ODS Graphics Produced by PROC UCM

ODS Graph Name	Plot Description	PLOT= Option
	ODS Plots Related to the Residual Analysis	
ErrorACFPlot	Prediction Error Autocorrelation Plot. ESTIMATE statement.	PLOT=ACF
ErrorHistogram	Prediction Error Histogram. ESTIMATE statement.	PLOT=NORMAL
ErrorPlot	Plot of Prediction Errors. ESTIMATE statement.	PLOT=RESIDUAL
	ODS Plots Related to the Series Forecasts	
ForecastsOnlyPlot	Series Forecasts Beyond the Historical Period. FORECAST statement.	DEFAULT
ModelForecastsPlot	One-Step-Ahead as well as Multi-Step-Ahead Forecasts. FORECAST statement.	PLOT=FORECASTS
	ODS Plots Related to the Individual Components	
FilteredAutoregPlot	Plot of Filtered Autoreg Component. AUTOREG statement.	PLOT=FILTER
FilteredBlockSeasonPlot	Plot of Filtered Block Season Component. BLOCKSEASON statement.	PLOT=FILTER
FilteredCyclePlot	Plot of Filtered Cycle Component. CYCLE statement.	PLOT=FILTER
FilteredIrregularPlot	Plot of Filtered Irregular Component. IRREGULAR statement.	PLOT=FILTER

Table 29.2. (continued)

ODS Graph Name	Plot Description	Option
FilteredLevelPlot	Plot of Filtered Level Component. LEVEL statement.	PLOT=FILTER
FilteredSeasonPlot	Plot of Filtered Season Component. SEASON statement.	PLOT=FILTER
FilteredSlopePlot	Plot of Filtered Slope Component. SLOPE statement.	PLOT=FILTER
SmoothedAutoregPlot	Plot of Smoothed Autoreg Component. AUTOREG statement.	PLOT=SMOOTH
SmoothedBlockSeasonPlot	Plot of Smoothed Block Season Component. BLOCKSEASON statement.	PLOT=SMOOTH
SmoothedCyclePlot	Plot of Smoothed Cycle Component. CYCLE statement.	PLOT=SMOOTH
SmoothedIrregularPlot	Plot of Smoothed Irregular Component. IRREGULAR statement.	PLOT=SMOOTH
SmoothedLevelPlot	Plot of Smoothed Level Component. LEVEL statement.	PLOT=SMOOTH
SmoothedSeasonPlot	Plot of Smoothed Season Component. SEASON statement.	PLOT=SMOOTH
SmoothedSlopePlot	Plot of Smoothed Slope Component. SLOPE statement.	PLOT=SMOOTH
	ODS Plots Related to the Series Decomposition	
FilteredAllExceptIrregPlot	Plot of Sum of All Filtered Components Except the Irregular. FORECAST statement.	PLOT= FDECOMP
FilteredTrendPlot	Plot of Filtered Trend. FORECAST statement.	PLOT= FDECOMP
FilteredTrendRegCycPlot	Plot of Sum of Filtered Trend, Cycles and Regression Effects. FORECAST statement.	PLOT= FDECOMP
FilteredTrendRegPlot	Plot of Filtered Trend Plus Regression Effects. FORECAST statement.	PLOT= FDECOMP
SmoothedAllExceptIrregPlot	Plot of Sum of All Smoothed Components Except the Irregular. FORECAST statement.	PLOT= DECOMP
SmoothedTrendPlot	Plot of Smoothed Trend. FORECAST statement.	PLOT= TREND
SmoothedTrendRegCycPlot	Plot of Sum of Smoothed Trend, Cycles and Regression Effects. FORECAST statement.	PLOT= DECOMP

Table 29.2. (continued)

ODS Graph Name	Plot Description	Option
SmoothedTrendRegPlot	Plot of Smoothed Trend Plus Regression Effects. FORECAST statement.	PLOT= DECOMP

OUTFOR= Data Set

You can use OUTFOR= option in the FORECAST statement to store the series and component forecasts produced by the procedure. This data set contains the following columns:

- the BY variables
- the ID variable. If an ID variable is not specified then a numerical variable, _ID_, is created that contains the observation numbers from the input data set.
- the dependent series and the predictor series
- FORECAST, a numerical variable containing the one step ahead predicted values and the multi step forecasts
- RESIDUAL, a numerical variable containing the difference between the actual and forecast values
- STD, a numerical variable containing the standard error of prediction
- LCL and UCL, numerical variables containing the lower and upper forecast confidence limits
- S_SERIES and VS_SERIES, numerical variables containing the smoothed values of the dependent series and their variances
- S_IRREG and VS_IRREG, numerical variables containing the smoothed values of the IRREGULAR component and their variances. These variables are present only if the model has an IRREGULAR component.
- F_LEVEL, VF_LEVEL, S_LEVEL, and VS_LEVEL, numerical variables containing the filtered and smoothed values of the LEVEL component and the respective variances. These variables are present only if the model has a LEVEL component.
- F_SLOPE, VF_SLOPE, S_SLOPE, and VS_SLOPE, numerical variables containing the filtered and smoothed values of the SLOPE component and the respective variances. These variables are present only if the model has a SLOPE component.
- F_AUTOREG, VF_AUTOREG, S_AUTOREG, and VS_AUTOREG, numerical variables containing the filtered and smoothed values of the AUTOREG component and the respective variances. These variables are present only if the model has an AUTOREG component.
- F_CYCLE, VF_CYCLE, S_CYCLE, and VS_CYCLE, numerical variables containing the filtered and smoothed values of the CYCLE component and the respective variances. If there are multiple cycles in the model, these variables

are sequentially numbered as F_CYCLE1, F_CYCLE2, etc. These variables are present only if the model has at least one CYCLE component.

- F_SEASON, VF_SEASON, S_SEASON, and VS_SEASON, numerical variables containing the filtered and smoothed values of the SEASON component and the respective variances. If there are multiple seasons in the model, these variables are sequentially numbered as F_SEASON1, F_SEASON2, etc. These variables are present only if the model has at least one SEASON component.
- F_BLKSEAS, VF_BLKSEAS, S_BLKSEAS, and VS_BLKSEAS, numerical variables containing the filtered and smoothed values of the BLOCKSEASON component and the respective variances. If there are multiple block seasons in the model, these variables are sequentially numbered as F_BLKSEAS1, F_BLKSEAS2, etc. These variables are present only if the model has at least one BLOCKSEASON component.
- S_TREG and VS_TREG, numerical variables containing the smoothed values of level plus regression component and their variances. These variables are present only if the model has at least one predictor variable or has dependent lags.
- S_TREGCYC and VS_TREGCYC, numerical variables containing the smoothed values of level plus regression plus cycle component and their variances. These variables are present only if the model has at least one cycle or an autoreg component.
- S_NOIRREG and VS_NOIRREG, numerical variables containing the smoothed values of the sum of all components except the irregular component and their variances. These variables are present only if the model has at least one season or block season component.

OUTEST= Data Set

You can use OUTEST= option in the ESTIMATE statement to store the model parameters and the related estimation details. This data set contains the following columns:

- the BY variables
- COMPONENT, a character variable containing the name of the component corresponding to the parameter being described
- PARAMETER, a character variable containing the parameter name
- TYPE, a character variable indicating whether the parameter value was FIXED by the user or it was ESTIMATED
- _STATUS_, a character variable indicating whether the parameter estimation process Converged, Failed, or there was an Error of some other kind.
- ESTIMATE, a numerical variable containing the parameter estimate
- STD, a numerical variable containing the standard error of the parameter estimate. This will have a missing value if the parameter value is fixed.

- TVALUE, a numerical variable containing the *t*-statistic. This will have a missing value if the parameter value is fixed.
- PVALUE, a numerical variable containing the *p*-value. This will have a missing value if the parameter value is fixed.

Statistics of Fit

This section explains the goodness-of-fit statistics reported to measure how well the specified model fits the data.

First the various statistics of fit that are computed using the prediction errors, $y_t - \hat{y}_t$, are considered. In these formulae, *n* is the number of non-missing prediction errors and *k* is the number of fitted parameters in the model. Moreover, the sum of square errors, $SSE = \sum (y_t - \hat{y}_t)^2$, and the total sum of squares for the series corrected for the mean, $SST = \sum (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean and the sums are over all the non-missing prediction errors.

Mean Square Error
The mean squared prediction error, MSE, calculated from the one step ahead forecasts. $MSE = \frac{1}{n} SSE$.

Root Mean Square Error
The root mean square error (RMSE), $\sqrt{MSE}$.

Mean Absolute Percent Error
The mean absolute percent prediction error (MAPE), $\frac{100}{n} \sum_{t=1}^{n} |(y_t - \hat{y}_t)/y_t|$.
The summation ignores observations where $y_t = 0$.

R-Square
The R^2 statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares, *SSE*, may be larger than *SST* and the R^2 statistic will be negative.

Adjusted R-Square
The adjusted R^2 statistic, $1 - (\frac{n-1}{n-k})(1 - R^2)$.

Amemiya's Adjusted R-Square
Amemiya's adjusted R^2, $1 - (\frac{n+k}{n-k})(1 - R^2)$.

Random Walk R-Square
The random walk R^2 statistic (Harvey's R^2 statistic using the random walk model for comparison), $1 - (\frac{n-1}{n}) SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$, and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$.

Maximum Percent Error
The largest percent prediction error, $100 \max((y_t - \hat{y}_t)/y_t)$. In this computation the observations where $y_t = 0$ are ignored.

The likelihood-based fit statistics are reported separately. They include the full log-likelihood, the diffuse part of the log-likelihood (see "Details"), the normalized residual sum of squares, and the Akaike and Bayesian information criteria. Let L denote the log-likelihood, k denote the sum of number of estimated parameters and the number of diffuse elements in the state vector, and n be the number of non-missing measurements in the estimation span. Moreover, let d denote the initialization period, and, ν_t and F_t denote the one step ahead prediction errors and their variances.

Normalized Residual Sum of Squares
Normalized Residual Sum of Squares, $\sum_{t=d+1}^{n} \frac{\nu_t^2}{F_t}$.

Akaike's Information Criterion
Akaike's information criterion (AIC), $-2L + 2k$.

Schwarz Bayesian Information Criterion
Schwarz Bayesian information criterion (SBC or BIC),
$-2L + k \log(n)$.

Examples

Example 29.1. An Example of a BSM

The series in this example, the monthly Airline Passenger series, has already been discussed in an example in the Getting Started section, see "A Seasonal Series with a Linear Trend". Recall that the series consists of monthly numbers of international airline travelers (from January 1949 to December 1960). Here we will examine additional output and use the ESTIMATE and FORECAST statements to limit the span of the data used in parameter estimation and forecasting. The following syntax fits a BSM to the logarithm of the airline passenger numbers. The disturbance variance for the SLOPE component is held fixed at value 0; that is, the trend is locally linear with constant slope. In order to evaluate the performance of the fitted model on observed data, some of the observed data are withheld during parameter estimation and forecast computations. The observations in the last two years, years 1959 and 1960, are not used in parameter estimation while the observations in the last year, year 1960, are not used in the forecasting computations. This is done using the BACK= option in the ESTIMATE and FORECAST statements.

```
   data series_g;
        set sashelp.air;
        logair = log(air);
   run;


proc ucm data = series_g;
     id date interval = month;
     model logair;
     irregular;
```

```
         level;
         slope var = 0 noest;
         season length = 12 type=trig;
         estimate back=24;
         forecast back=12 lead=24 print=forecasts;
    run;
```

The following tables display the summary of data used in estimation and forecasting (Output 29.1.1 and Output 29.1.2). These tables provide simple summary statistics for the estimation and forecast spans; they include useful information such as the start and end dates of the span, number of non-missing values, etc.

Output 29.1.1. Observation Span Used in Parameter Estimation (partial output)

```
                          Estimation Span Summary

     Variable      Type         First       Last      Nobs           Mean

      logair     Dependent     JAN1949     DEC1958      120        5.43035
```

Output 29.1.2. Observation Span Used in Forecasting (partial output)

```
                           Forecast Span Summary

     Variable      Type         First       Last      Nobs           Mean

      logair     Dependent     JAN1949     DEC1959      132        5.48654
```

The following tables display the fixed parameters in the model, the preliminary estimates of the free parameters, and the final estimates of the free parameters (Output 29.1.3, Output 29.1.4, and Output 29.1.5).

Output 29.1.3. Fixed Parameters in the Model

```
                            The UCM Procedure

                      Fixed Parameters in the Model

                 Component     Parameter               Value

                 Slope         Error Variance              0
```

Output 29.1.4. Starting Values for the Parameters to Be Estimated

```
                              Preliminary  Estimates
                              of the Free Parameters

                    Component     Parameter            Estimate

                    Irregular     Error Variance        6.64120
                    Level         Error Variance        2.49045
                    Season        Error Variance        1.26676
```

Output 29.1.5. Maximum Likelihood Estimates of the Free Parameters

```
                     Final Estimates of the Free Parameters

                                                    Approx                  Approx
Component     Parameter             Estimate     Std Error    t Value     Pr > |t|

Irregular     Error Variance      0.00018686     0.0001212       1.54       0.1233
Level         Error Variance      0.00040314     0.0001566       2.57       0.0100
Season        Error Variance      0.00000350    1.66319E-6       2.10       0.0354
```

Two types of goodness of fit statistics are reported after a model is fit to the series (see Output 29.1.6 and Output 29.1.7). The first type is the likelihood-based goodness of fit statistics, which include the full likelihood of the data, the diffuse portion of the likelihood (see "Details"), and the Akike and Bayesian Information criteria. The second type of statistics is based on the raw residuals, residual = observed - predicted. If the model is non-stationary, then one step ahead predictions are not available for some initial observations, and the number of values used in computing these fit statistics will be different from those used in computing the likelihood-based test statistics.

Output 29.1.6. Likelihood Based Fit Statistics for the Airline Data

```
                          Likelihood Based Fit Statistics

                Full Log-Likelihood                    168.67997
                Diffuse Part of Log-Likelihood         -13.92861
                Normalized Residual Sum of Squares     107.00000
                Akaike Information Criterion          -305.35994
                Bayesian Information Criterion        -260.76007

                     Number of non-missing observations used
                     for computing the log-likelihood = 120
```

Output 29.1.7. Residuals Based Fit Statistics for the Airline Data

```
                        Fit Statistics Based on Residuals

                  Mean Squared Error                      0.00156
                  Root Mean Squared Error                 0.03944
                  Mean Absolute Percentage Error          0.57677
                  Maximum Percent Error                   2.19396
                  R-Square                                0.98705
                  Adjusted R-Square                       0.98680
                  Random Walk R-Square                    0.99315
                  Amemiya’s Adjusted R-Square             0.98630

                    Number of non-missing residuals used for
                       computing the fit statistics = 107
```

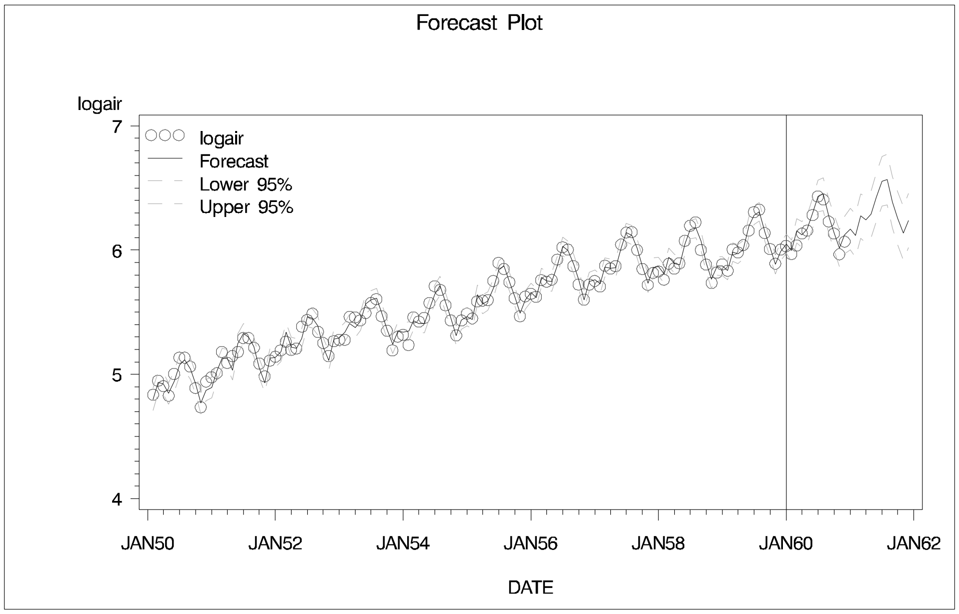
The forecasts are given in Output 29.1.8. In order to save the space, the upper and lower confidence limit columns are dropped from the output, and only the rows corresponding to the year 1960 are shown. Recall that the actual measurements in the years 1959 and 1960 were withheld during the parameter estimation, and the ones in 1960 were not used in the forecast computations.

Output 29.1.8. Forecasts for the Airline Data

```
                   Forecasts for Airline Data (partial output)

     Obs       date       Forecast       StdErr         logair       Residual

     133    JAN1960    6.049900848    0.0383865    6.033086222    -0.01681463
     134    FEB1960    5.996181814     0.043925     5.96870756    -0.02747425
     135    MAR1960     6.15571288    0.0492542     6.03787092    -0.11784196
     136    APR1960    6.123514784    0.0534002    6.133398043    0.009883259
     137    MAY1960    6.168045435    0.0576337    6.156978986    -0.01106645
     138    JUN1960    6.302872975    0.0610994    6.282266747    -0.02060623
     139    JUL1960    6.434621832    0.0646732    6.432940093    -0.00168174
     140    AUG1960    6.449565514    0.0676591    6.406879986    -0.04268553
     141    SEP1960    6.265131851    0.0706778    6.230481448     -0.0346504
     142    OCT1960    6.138451548    0.0731438    6.133398043    -0.00505351
     143    NOV1960    6.015324248    0.0754033    5.966146739    -0.04917751
     144    DEC1960    6.121205238    0.0768566    6.068425588    -0.05277965
```

The figure Output 29.1.9 shows the forecast plot. The forecasts in the year 1960 show that the model predictions were quite good.

Output 29.1.9. Forecast Plot of the Airline Series Using a BSM



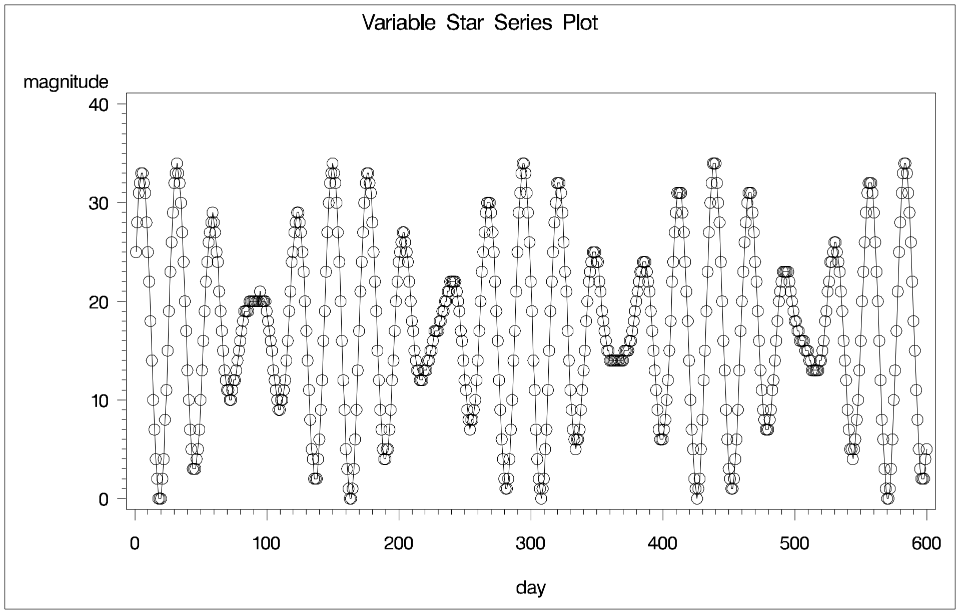
Example 29.2. Variable Star Data

The series in this example is studied in detail in Bloomfield (2000). This series consists of brightness measurements (magnitude) of a variable star taken at midnight for 600 consecutive days. The data can be downloaded from a time series archive maintained by the University of York, England (http://www.york.ac.uk/depts/maths/data/ts/welcome.htm (series number 26)). The following DATA step statements read the data in a SAS data set.

```
data star;
    input magnitude @@;
    datalines;

        25 28 31 32 33 33 32 31 28 25
        22 18 14 10  7  4  2  0  0  0
         2  4  8 11 15 19 23 26 29 32

        /* -- data lines removed - */

        31 33 34 34 33 31 29 26 22 18
        15 11  8  5  3  2  2  2  4  5
     ;
```

The plot of the series is shown in figure Output 29.2.1.

Output 29.2.1. Plot of Star Brightness on Successive Days



The plot clearly shows the cyclic nature of the series. Bloomfield shows that the series is very well explained by a model that includes two deterministic cycles that have periods 29.0003 and 24.0001 days, a constant term, and a simple error term. He also shows the difficulty involved in estimating the periods from the data (see Bloomfield 2000, Chapter 3). The parameters are estimated by the least squares, and the sum of squares surface has multiple local optima and ridges. In this example we will use the UCM procedure to model this series. We begin with a model that consists of only one stochastic cycle and the level and irregular components. This is because it is quite possible that a single stochastic cycle with time varying amplitude and phase may be adequate to explain the observed series.

```
proc ucm data=star;
    model magnitude;
    irregular;
    level;
    cycle;
    forecast print=forecasts;
run;
```

The final parameter estimates and the goodness of fit statistics are shown below (see Output 29.2.2 and Output 29.2.3).

Output 29.2.2. Parameter Estimates in Single Cycle Model

```
                               The UCM Procedure

                      Final Estimates of the Free Parameters

                                                   Approx                   Approx
Component     Parameter            Estimate     Std Error     t Value     Pr > |t|

Irregular     Error Variance        0.02094     0.0076007        2.76       0.0059
Level         Error Variance     3.6126E-10     2.2014E-7        0.00       0.9987
Cycle         Damping Factor        0.99906     0.0007979     1252.18       <.0001
Cycle         Period               27.12640       0.17225      157.48       <.0001
Cycle         Error Variance        0.20111       0.16915        1.19       0.2345
```

Output 29.2.3. Model Fit of Single Cycle Model

```
                          Fit Statistics Based on Residuals

                  Mean Squared Error                     0.31481
                  Root Mean Squared Error                0.56108
                  Mean Absolute Percentage Error         4.55372
                  Maximum Percent Error                 60.57167
                  R-Square                               0.99609
                  Adjusted R-Square                      0.99607
                  Random Walk R-Square                   0.94510
                  Amemiya’s Adjusted R-Square            0.99603

                    Number of non-missing residuals used for
                       computing the fit statistics = 599
```

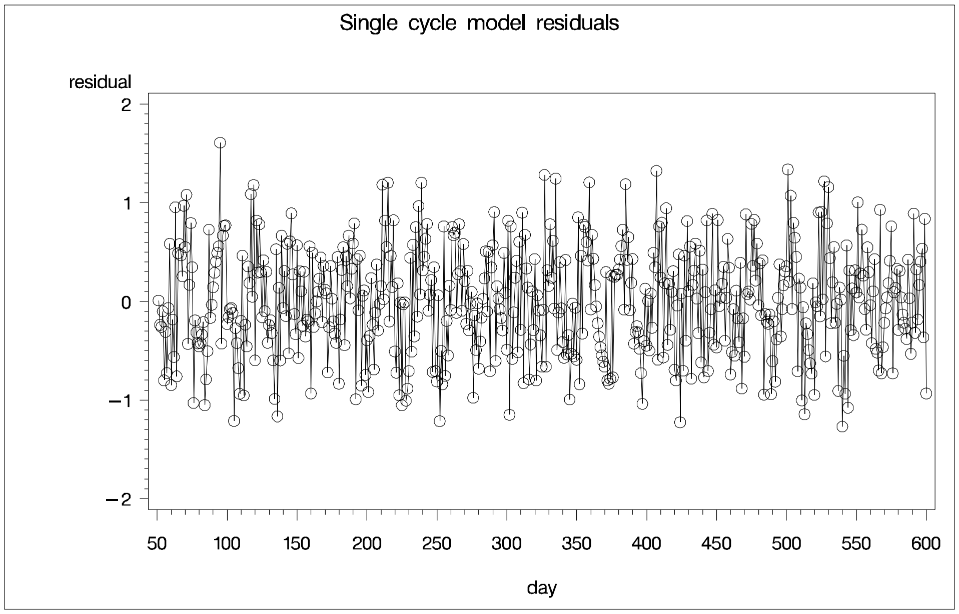
A description of the cycle in the model is given below (Output 29.2.4).

Output 29.2.4. Summary of the Cycle in the Single Cycle Model

```
                        Cycle description (partial output)

Name         Type             period            Rho         CycleVar         ErrorVar

Cycle    Stationary         27.12640        0.99906        106.68773          0.20111
```

From the model fit it appears that the single stochastic cycle model with a period of approximately 27 days fits the data reasonably well. However, the residual plot in Output 29.2.5 indicates that the series may contain an additional cycle. The autocorrelation plot of residuals shows this more clearly (this plot is not shown).

Output 29.2.5. Residual Plot for the Single Cycle Model



The following statements fit a two-cycle model to the series.

```
proc ucm data=star;
    model magnitude;
    irregular;
    level var=0 noest;
    cycle rho=1 noest=rho;
    cycle rho=1 noest=rho;
    forecast print=forecasts;
run;
```

The model has two cycles, both with damping factors of one, and a constant level component. The final parameter estimates and the goodness of fit statistics are shown below (see Output 29.2.6 and Output 29.2.7).

Output 29.2.6. Parameter Estimates in Two Cycle Model

```
                                  The UCM Procedure

                         Final Estimates of the Free Parameters

                                                          Approx                 Approx
Component    Parameter             Estimate            Std Error    t Value    Pr > |t|

Irregular    Error Variance         0.09189            0.0053285      17.24      <.0001
Cycle_1      Period                24.00010            0.0013342    17988.2      <.0001
Cycle_1      Error Variance      2.89893E-12           3.08743E-9      0.00      0.9993
Cycle_2      Period                29.00027            0.0013891    20877.1      <.0001
Cycle_2      Error Variance       2.7808E-12           3.00071E-9      0.00      0.9993
```

Output 29.2.7. Fit of Two Cycle Model

```
                 Fit Statistics Based on Residuals

                  Mean Squared Error                  0.19206
                  Root Mean Squared Error             0.43825
                  Mean Absolute Percentage Error      2.72498
                  Maximum Percent Error              39.03001
                  R-Square                            0.99759
                  Adjusted R-Square                   0.99758
                  Random Walk R-Square                0.96935
                  Amemiya's Adjusted R-Square         0.99755

                    Number of non-missing residuals used for
                       computing the fit statistics = 595
```

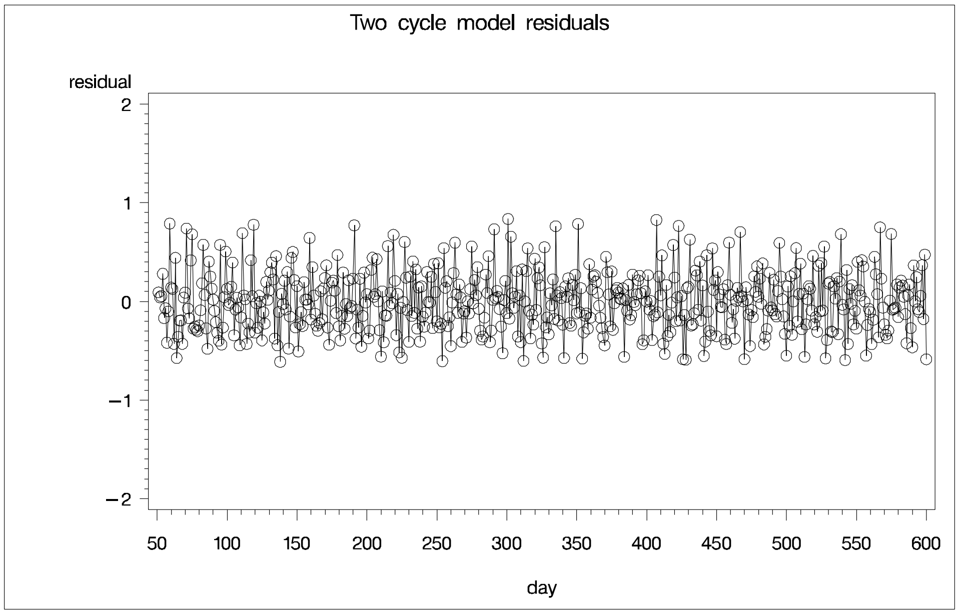
A description of the cycles in the model is given in Output 29.2.8.

Output 29.2.8. Summary of the Cycles in the Two Cycle Model

```
                 Cycle description (partial output)

       Name          Type               period          Rho       ErrorVar

      Cycle_1    Non-Stationary       24.00010      1.00000    2.89893E-12
      Cycle_2    Non-Stationary       29.00027      1.00000     2.7808E-12
```

Note that the estimated periods are the same as Bloomfield's model, and the disturbance variances are very close to zero, implying deterministic cycles. In fact, this model is identical to Bloomfield's model. The residual plot shown below also shows (Output 29.2.9) the improvement in the fit; for easy comparison the scale of the vertical axis is purposely set to be the same as the residual plot of the single-cycle model.

Output 29.2.9. Residual Plot for the Two Cycle Model



Example 29.3. Modeling Long Seasonal Patterns

In this example the use of the UCM procedure to model complex seasonal patterns is illustrated. Only a broad outline of the analysis is presented. The time series used in this example consists of number of calls received per shift at a call center. Each shift is of a three-hour duration, and the first shift of the day begins at midnight, resulting in eight shifts per day. The observations are available from December 15, 1999, up to April 30, 2000. Initial exploration of the series clearly shows that the time of the day and the day of the week have a significant influence on the number of calls received; however, the series does not show any significant trend. This suggests a simple random walk trend model along with a seasonal component with season length of 56. After fitting this model and examining the residual plots (not shown), the following modified model seems to fit the data quite well.

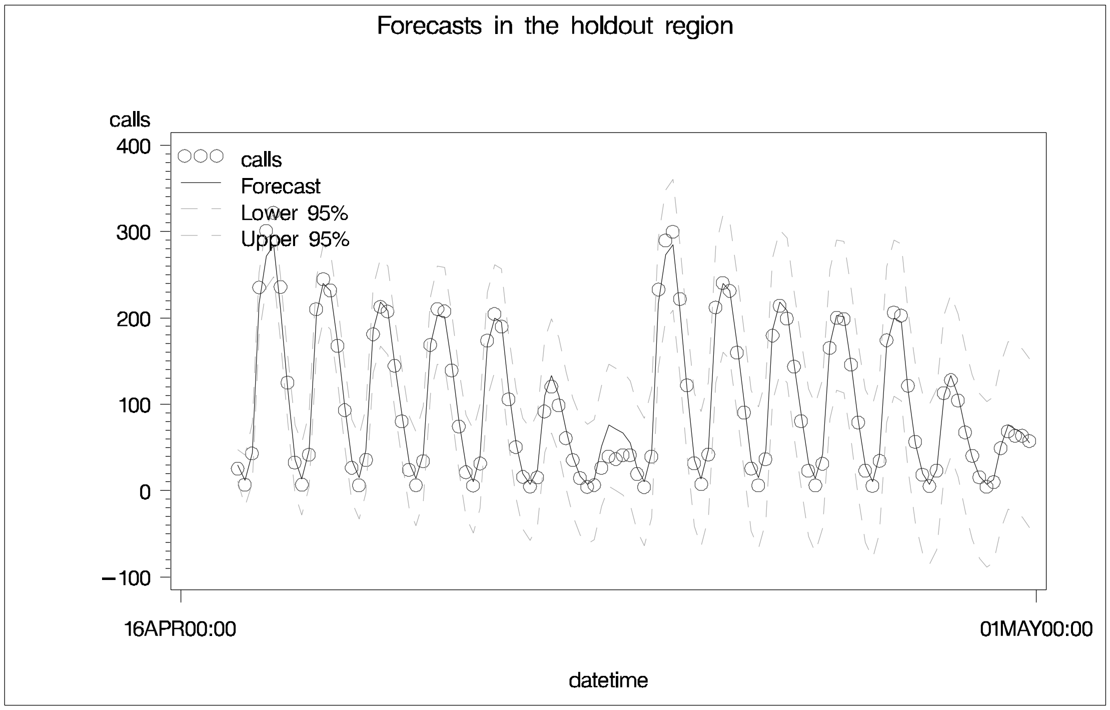
```
proc ucm data=callcenter;
    id datetime interval=dthour3;
    model calls;
    irregular;
    level;
    season length=56 type=trig;
    deplag lags=2;
    estimate back=112;
    forecast back=112 lead = 112;
run;
```

Along with the seasonal component of length 56 and the level and irregular components, this model also includes two lags of the dependent variable. The last two weeks of the data are withheld to examine the forecasting performance of the model on the observed data. The table in Output 29.3.1 shows the fit statistics for this model, and the plot in Output 29.3.2 shows the fit of the model on the withheld data.

Output 29.3.1. Model Fit for the Saturated Seasonal Model

```
                         The UCM Procedure

                  Fit Statistics Based on Residuals

         Mean Squared Error                        93.91316
         Root Mean Squared Error                    9.69088
         Mean Absolute Percentage Error            19.75413
         Maximum Percent Error                   1102.00394
         R-Square                                   0.98664
         Adjusted R-Square                          0.98666
         Random Walk R-Square                       0.97575
         Amemiya’s Adjusted R-Square                0.98664

           Number of non-missing residuals used for
              computing the fit statistics = 934
```

Output 29.3.2. Forecasts for the Saturated Seasonal Model



This model fits the data well; however, it takes a considerable amount of time to estimate because the dimension of the state vector in the underlying state space model is large, about 60 (see “Details”). This problem becomes more acute for larger season lengths. More importantly, this model does not offer any insight into the structure of the particular seasonal pattern present at this call center; for example, it may be possible that a similar calling pattern may repeat during each day of the week except for different mean number of calls for different days of the week. Such a pattern is implied by a combination of two components, a simple seasonal of season length 8, for capturing the within day variation, and a block seasonal component of 7 blocks (a block for each day of the week) with block size of 8 that models the day to day variation during the week. The state vector of this model has dimension 17. Examination of the fit of this model reveals that the model fit is poorer than the above saturated seasonal model. The forecasts for Sundays and Mondays are particularly bad. This behavior can be improved somewhat by including dummy variables corresponding to some of the shifts on these days. The following statements specify one such model.

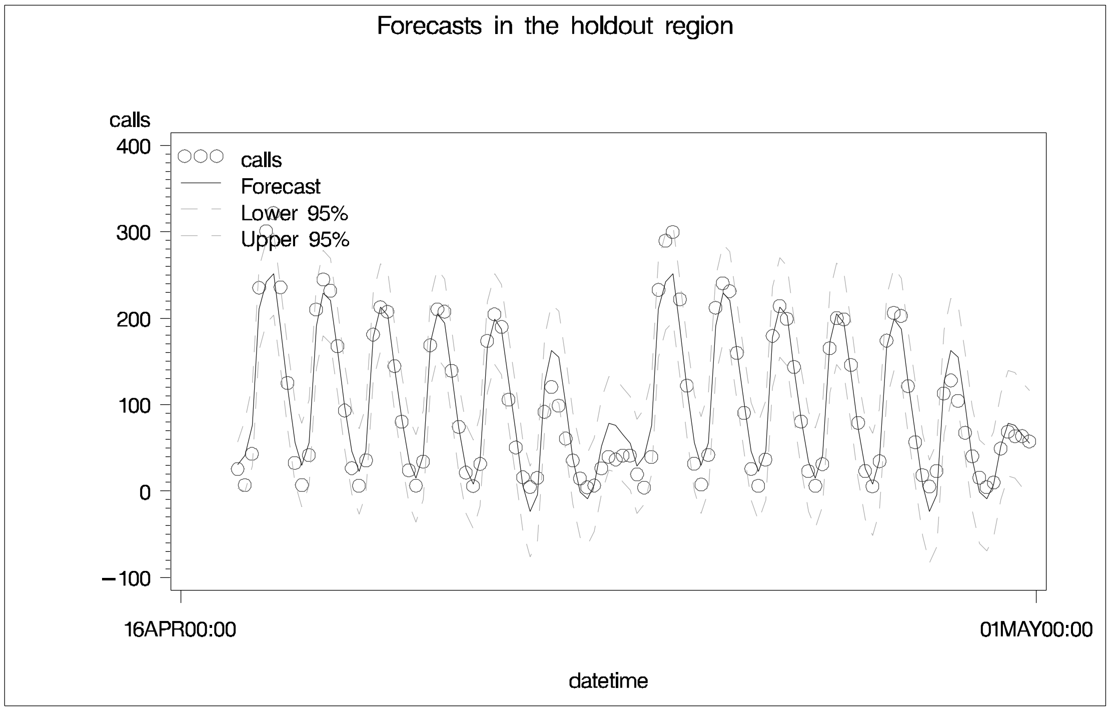
```
proc ucm data=callcenter;
    id datetime interval=dthour3;
    model calls = s_shift3 s_shift4 s_shift6
                  m_shift1 m_shift3 m_shift6;
    irregular;
    level;
    season length=8 type=trig;
    blockseason nblocks=7 blocksize=8 type=trig;
    deplag lags=2;
    estimate back=112;
    forecast back=112 lead = 112 print=forecasts;
run;
```

The model contains dummy regressors for three shifts on Sundays, shifts 3, 4, and 6, and three shifts on Mondays, shifts 1, 3, and 6. The table in Output 29.3.3 shows the fit statistics for this model, and the plot in Output 29.3.4 shows the fit of the model on the withheld data.

Output 29.3.3. Model Fit for the Block Seasonal Model

```
                                The UCM Procedure

                        Fit Statistics Based on Residuals

                 Mean Squared Error                   183.66579
                 Root Mean Squared Error               13.55234
                 Mean Absolute Percentage Error        38.34285
                 Maximum Percent Error                447.05942
                 R-Square                               0.97387
                 Adjusted R-Square                      0.97373
                 Random Walk R-Square                   0.95248
                 Amemiya's Adjusted R-Square            0.97354

                   Number of non-missing residuals used for
                      computing the fit statistics = 936
```

Output 29.3.4. Forecasts for the Block Seasonal Model



This model can be improved further using similar considerations. The use of block seasonals is only one of several ways to model seasonals with long season lengths. Alternatively, you can also use a smaller set of harmonics than the full set used in the saturated model. This is done by specifying a separate cycle statement for each of the harmonics included in the model.

Example 29.4. Illustration of ODS Graphics (Experimental)

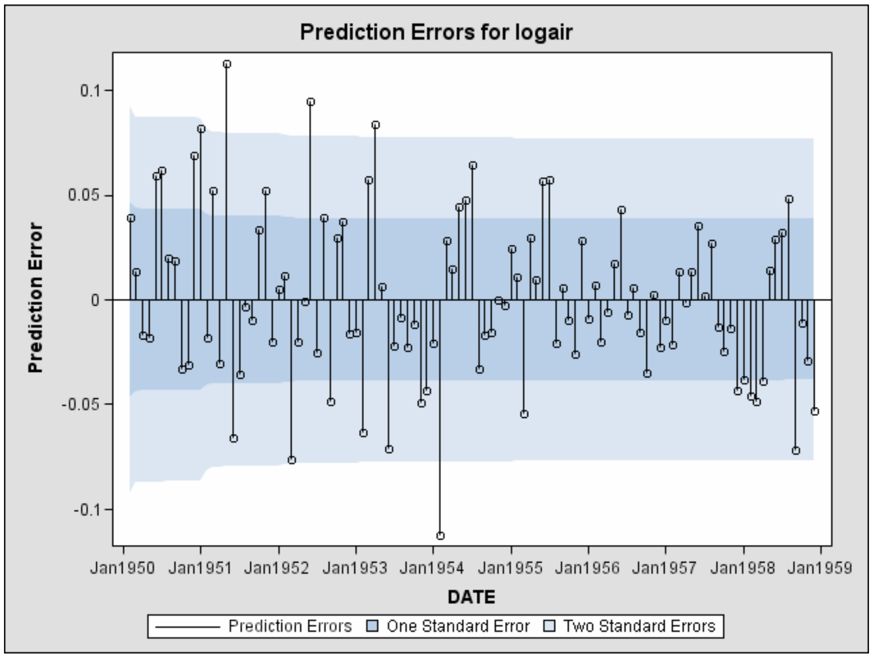
This example illustrates the use of experimental ODS graphics. The graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the UCM procedure, see the "ODS Graphics" section on page 1664.

The following code shows how you can use the PLOT= option in different statements of the UCM procedure to get useful plots for the Airline Passenger example discussed earlier; see Example 29.1. The PLOT=SMOOTH option in the SEASON statement requests plotting of the smoothed estimate of that seasonal component. The use of PLOT=(RESIDUAL NORMAL ACF) in the ESTIMATE statement produces the time series plot of residuals, the histogram of residuals, and the autocorrelation plot of residuals, respectively. Finally, the use of PLOT=(FORECASTS DECOMP) option in the FORECAST statement produces the forecast, the trend, and the trend plus season plots.

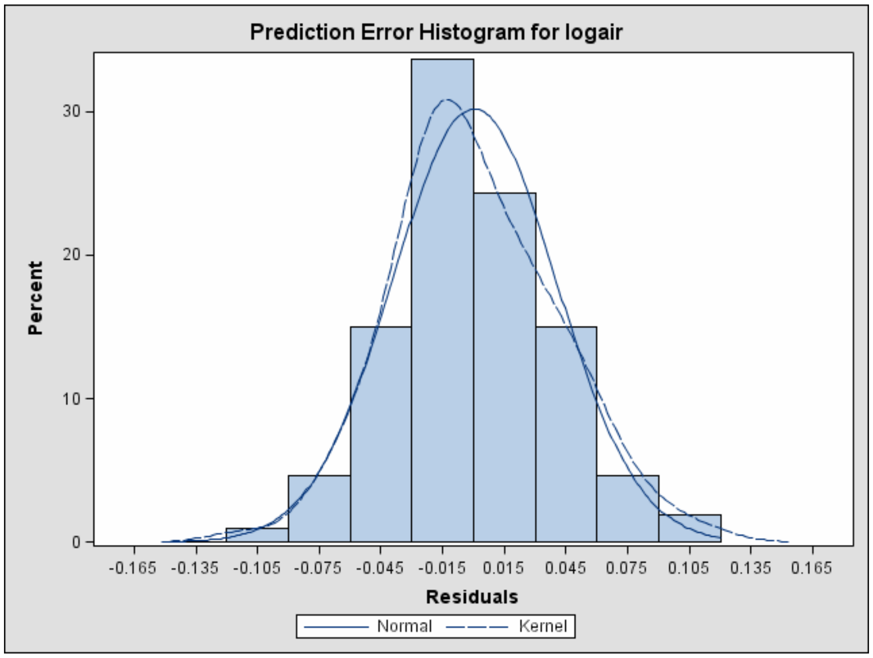
```
ods html;
ods graphics on;

proc ucm data=series_g;
   id date interval=month;
   model logair;
   irregular;
   level;
   slope variance=0 noest;
   season length=12 type=trig plot=smooth;
   estimate back=24 plot=(residual normal acf);
   forecast back=24 lead=36 plot=(forecasts decomp);
run;

ods graphics off;
ods html close;
```

Output 29.4.1 through Output 29.4.7 show a selection of the plots created.

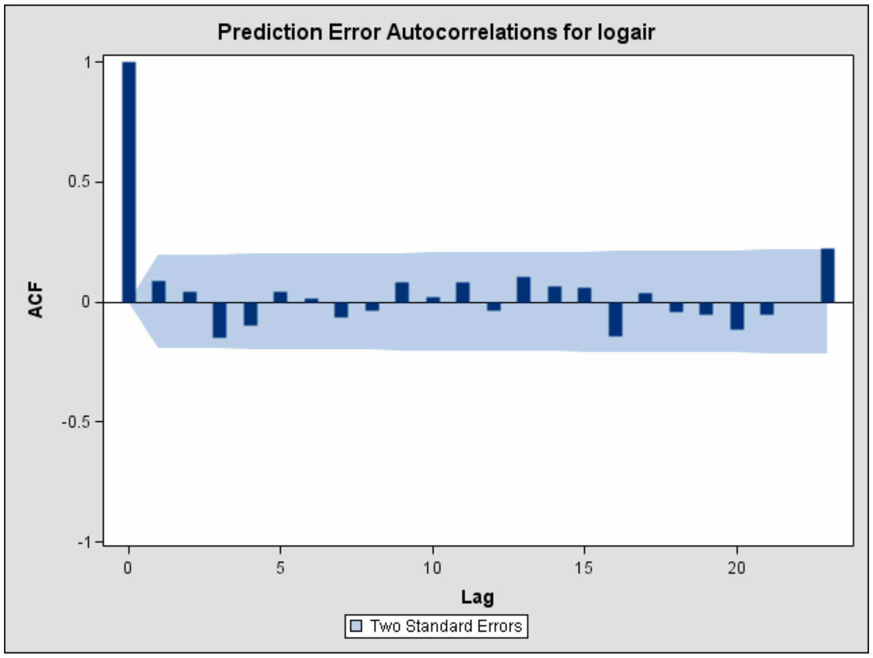
Output 29.4.1. Residual Plot (Experimental)



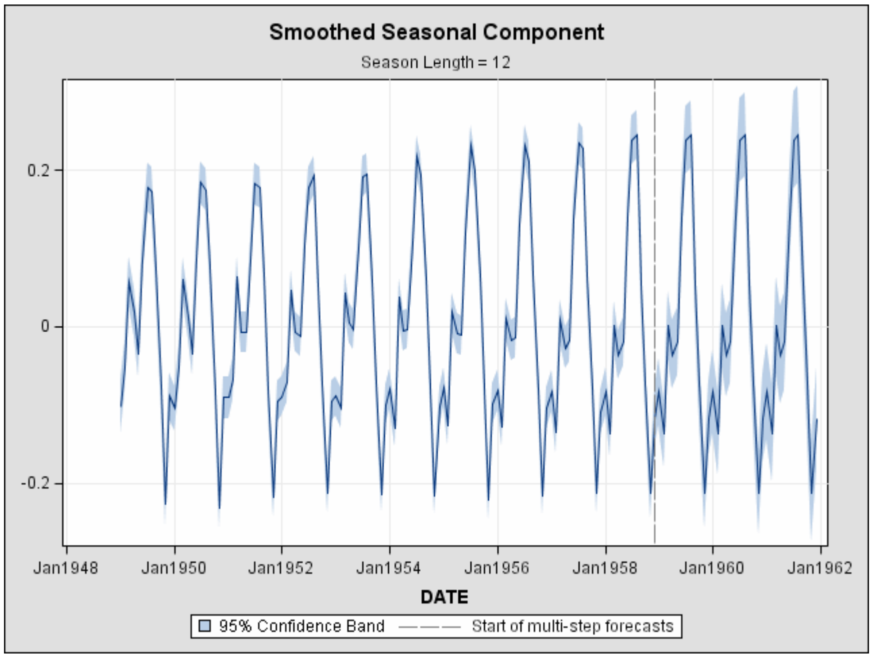
Output 29.4.2. Residual Histogram (Experimental)



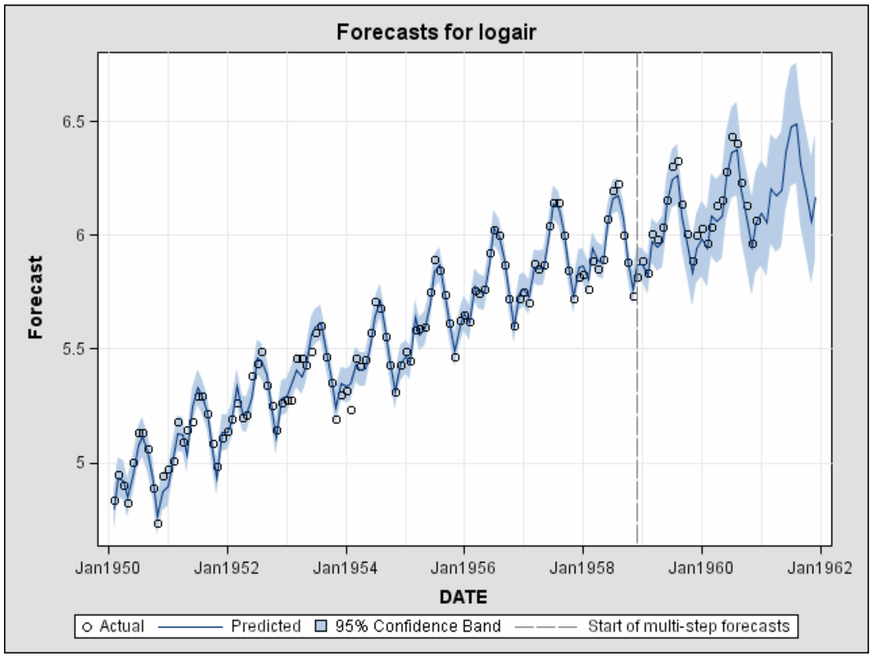
Output 29.4.3. Residual Autocorrelations (Experimental)



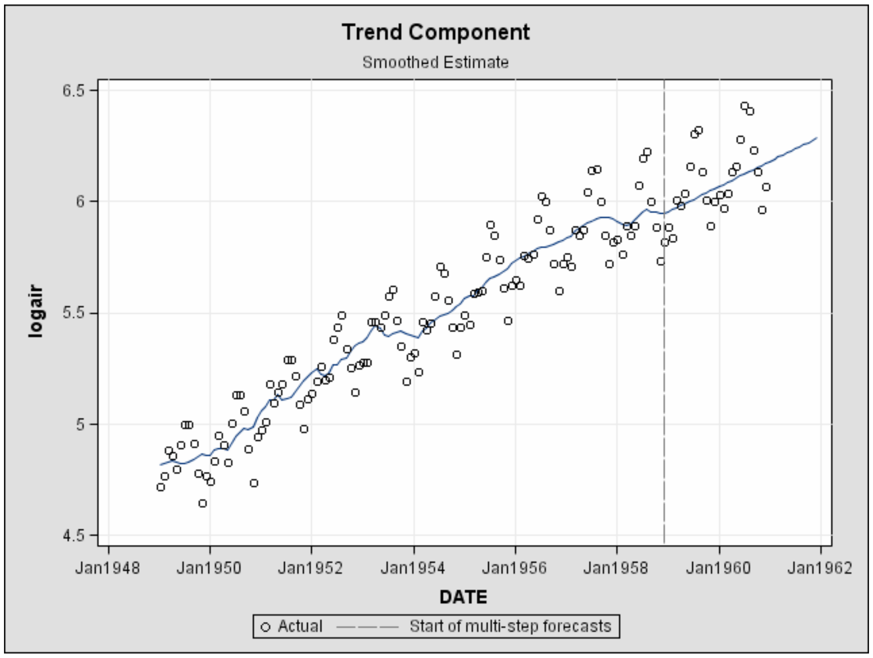
Output 29.4.4. Smoothed Seasonal (Experimental)



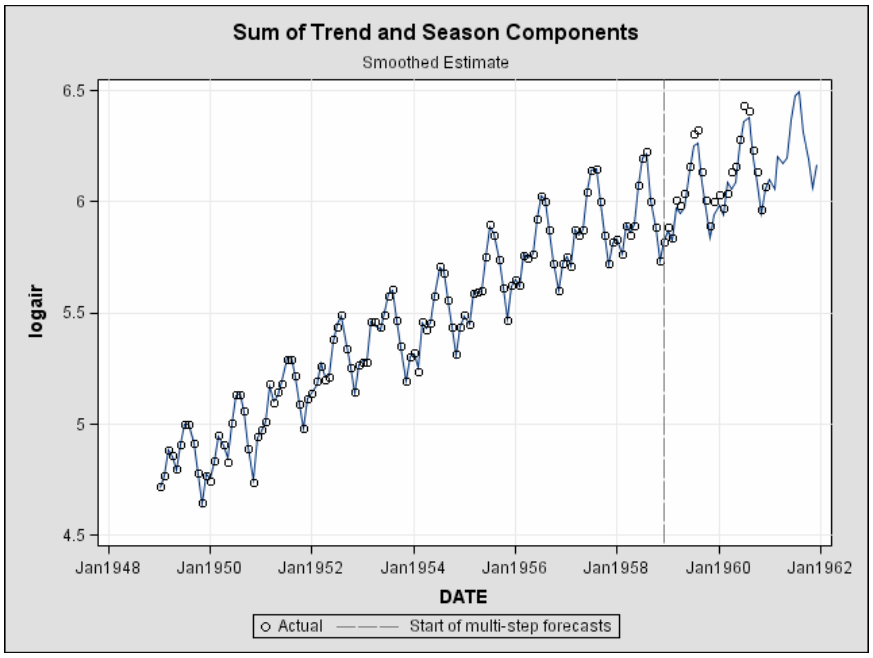
Output 29.4.5. Series Forecasts (Experimental)



Output 29.4.6. Trend Plot (Experimental)



Output 29.4.7. Trend Plus Season (Experimental)



Example 29.5. Many Syntax Illustrations

The following code fragments illustrate the PROC UCM syntax for some of the commonly needed modeling activities.

```
/* Dependent series, sales, is modeled using two predictor
   series, promo1 and promo2.  The data are quarterly.
   The forecasts are computed for twelve periods in the
   future.  All printing is suppressed.  The series and
   component forecasts are stored in an output data set,
   f_out.  The parameter estimates are stored in e_out;
*/
proc ucm data=company_x noprint;
     id date interval=qtr;
     model sales = promo1 promo2;
     irregular;
     level;
     estimate outest=e_out;
     forecast lead=12 outfor=f_out;
run;


/* Request printing of the filtered and smoothed seasonal
   component.
*/
proc ucm data=company_x;
     id date interval=qtr;
     model sales = promo1 promo2;
     irregular;
     level;
     season length=4 print=(filter smooth);
run;


/* Control the span of observations used in the estimation
   of model parameters using the SKIPFIRST= and BACK=
   options in the ESTIMATE statement.
*/
proc ucm data=company_x;
     id date interval=month;
     model sales = promo1 promo2;
     irregular;
     level;
     estimate skipfirst=10 back=12;
run;


/* Supply starting values for parameters. */
proc ucm data=company_x;
     model sales;
     irregular;
     level variance=10.3;
     deplag lags=2 phi=0.2 -1.8;
run;
```

```
/* Fix parameter values */
proc ucm data=company_x;
     model sales;
     irregular;
     level variance=10.3 noest;
     cycle period=4 noest=period;
     deplag lags=2 phi=0.2 -1.8 noest;
run;


/* Using cycles to get an "unsaturated" seasonal model.
   A monthly seasonal model using only the first three
   harmonics.
*/
proc ucm data=company_x;
     model sales;
     irregular;
     level variance=10.3 noest;
     cycle period=12 rho=1 noest=(period rho);
     cycle period=6 rho=1 noest=(period rho);
     cycle period=4 rho=1 noest=(period rho);
run;
```

References

Anderson, T.W. (1971), *The Statistical Analysis of Time Series,* New York: John Wiley & Sons, Inc.

Bloomfield, P. (2000), *Fourier Analysis of Time Series, Second Edition,* New York: John Wiley & Sons, Inc.

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control,* San Francisco: Holden-Day.

Cobb, G.W. (1978), “The Problem of the Nile: Conditional Solution to a Change Point Problem,” *Biometrika,* 65, 243-251.

Durbin, J. and Koopman, S.J. (2001), *Time Series Analysis by State Space Methods,* Oxford: Oxford University Press.

Harvey, A.C. (1989), *Forecasting, Structural Time Series Models and the Kalman Filter,* Cambridge:Cambridge University Press.

Harvey, A.C. (2001), “Testing in Unobserved Components Models,” *Journal of Forecasting,* 20, 1-19.

West, M. and Harrison, J. (1999) *Bayesian Forecasting and Dynamic Models*, 2nd ed, New York: Springer-Verlag.

Chapter 30
The VARMAX Procedure

Chapter Contents

Chapter 30
The VARMAX Procedure

Overview

Given a multivariate time series, the VARMAX procedure estimates the model parameters and generates forecasts associated with Vector AutoRegressive Moving-Average processes with eXogenous regressors (VARMAX) models. Often, economic or financial variables are not only contemporaneously correlated to each other, they are also correlated to each other's past values. The VARMAX procedure can be used to model these types of time relationships. In many economic and financial applications, the variables of interest (dependent, response, or endogenous variables) are influenced by variables external to the system under consideration (independent, input, predictor, regressor, or exogenous variables). The VARMAX procedure enables you to model both the dynamic relationship between the dependent variables and between the dependent and independent variables.

VARMAX models are defined in terms of the orders of the autoregressive or moving-average process (or both). When you use the VARMAX procedure, these orders can be specified by options or they can be automatically determined. Criteria for automatically determining these orders include

- Akaike Information Criterion (AIC)
- Corrected AIC (AICC)
- Hannan-Quinn (HQ) Criterion
- Final Prediction Error (FPE)
- Schwarz Bayesian Criterion (SBC), also known as Bayesian Information Criterion (BIC)

If you do not wish to use the automatic order selection, the VARMAX procedure provides AR order identification aids:

- partial cross-correlations
- Yule-Walker estimates
- partial autoregressive coefficients
- partial canonical correlations

For situations where the stationarity of the time series is in question, the VARMAX procedure provides tests to aid in determining the presence of unit roots and cointegration. These tests include

- Dickey-Fuller tests

- Johansen cointegration test for nonstationary vector processes of integrated order one
- Stock-Watson common trends test for the possibility of cointegration among nonstationary vector processes of integrated order one
- Johansen cointegration test for nonstationary vector processes of integrated order two

For stationary vector times series (or nonstationary series made stationary by appropriate differencing), the VARMAX procedure provides for both Vector AutoRegressive (VAR) and Bayesian Vector AutoRegressive (BVAR) models. To cope with the problem of high dimensionality in the parameters of the VAR model, the VARMAX procedure provides both the Vector Error Correction Model (VECM) and Bayesian Vector Error Correction Model (BVECM). Bayesian models are used when prior information about the model parameters is available. The VARMAX procedure can allow independent (exogenous) variables with their distributed lags to influence dependent (endogenous) variables. The model parameter estimation methods are

- Least Squares (LS)
- Maximum Likelihood (ML)

The VARMAX procedure provides various hypothesis tests of long-run effects and adjustment coefficients using the likelihood ratio test based on Johansen cointegration analysis. The VARMAX procedure offers the likelihood ratio test of the weak exogeneity for each variable.

After fitting the model parameters, the VARMAX procedure provides for model checks and residual analysis using the following tests:

- Durbin-Watson (DW) statistics
- F test for autoregressive conditional heteroscedastic (ARCH) disturbance
- F test for AR disturbance
- Jarque-Bera normality test
- Portmanteau test

Forecasting is one of the main objectives of multivariate time series analysis. After successfully fitting the VAR, VARX, BVAR, VARMA, VECM, and BVECM model parameters, the VARMAX procedure computes predicted values based on the parameter estimates and the past values of the vector time series.

The VARMAX procedure supports several modeling features, including

- seasonal deterministic terms
- subset models
- multiple regression with distributed lags

- dead-start model that does not have present values of the exogenous variables
- GARCH-type multivariate conditional heteroscedasticity models

The VARMAX procedure provides a Granger-Causality test to determine the Granger-causal relationships between two distinct groups of variables. It also provides

- infinite order AR representation
- impulse response function (or infinite order MA representation)
- decomposition of the predicted error covariances
- roots of the characteristic functions for both the AR and MA parts to evaluate the proximity of the roots to the unit circle
- contemporaneous relationships among the components of the vector time series

Experimental graphics are now available with the VARMAX procedure. For more information, see the "ODS Graphics" section on page 1825.

Getting Started

This section outlines the use of the VARMAX procedure and gives five different examples of the kind of models supported.

Vector Autoregressive Process

Let $\mathbf{y}_t = (y_{1t}, \ldots, y_{kt})'$, $t = 1, 2, \ldots$, denote a k-dimensional time series vector of random variables of interest. The pth-order VAR process is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \cdots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

where the $\boldsymbol{\epsilon}_t$ is a vector white noise process with $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \ldots, \epsilon_{kt})'$ such that $\mathrm{E}(\boldsymbol{\epsilon}_t) = 0$, $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$, and $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s') = 0$ for $t \neq s$; $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)'$ is a constant vector and Φ_i is a $k \times k$ matrix.

Analyzing and modeling the series jointly enables you to understand the dynamic relationships over time among the series and to improve the accuracy of forecasts for individual series by utilizing the additional information available from the related series and their forecasts.

Example of Vector Autoregressive Model

Consider the first-order stationary vector autoregressive model

$$\mathbf{y}_t = \begin{pmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t, \quad \text{with } \Sigma = \begin{pmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{pmatrix}$$

The following IML procedure statements simulate a bivariate vector time series from this model to provide test data for the VARMAX procedure:

```
proc iml;
   sig = {1.0  0.5, 0.5 1.25};
   phi = {1.2 -0.5, 0.6 0.3};
   /* simulate the vector time series */
   call varmasim(y,phi) sigma = sig n = 100 seed = 34657;
   cn = {'y1' 'y2'};
   create simul1 from y[colname=cn];
   append from y;
quit;
```

The following statements plot the simulated vector time series $\mathbf{y}_t$ shown in Figure 30.1:

```
data simul1;
   set simul1;
   date = intnx( 'year', '01jan1900'd, _n_-1 );
   format date year4.;

proc gplot data=simul1;
   symbol1 v = none i = join l = 1;
   symbol2 v = none i = join l = 2;
   plot y1 * date = 1
        y2 * date = 2 / overlay;
run;
```

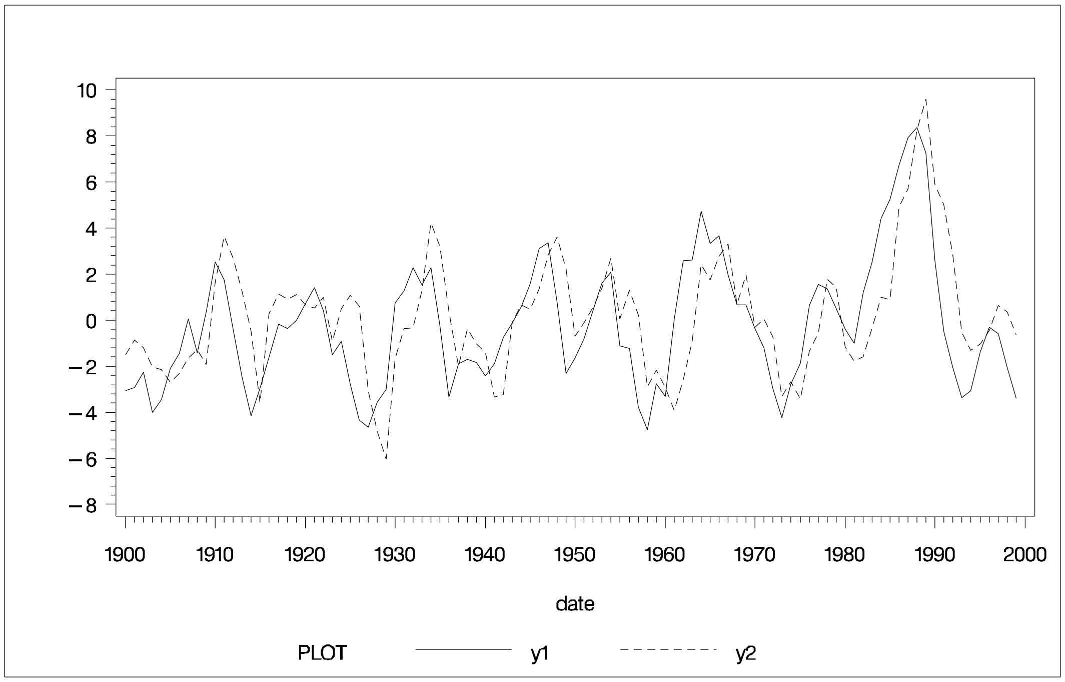


Figure 30.1. Plot of Generated Data Process

The following statements fit a VAR(1) model to the simulated data. You first specify the input data set in the PROC VARMAX statement. Then, you use the MODEL statement to read the time series y_1 and y_2. To estimate a VAR model with mean zero, you specify the order of the autoregressive model with the P= option and the NOINT option. The MODEL statement fits the model to the data and prints parameter estimates and various diagnostic tests. The LAGMAX=3 option is used to print the output for the residual diagnostic checks. For the forecasts, you specify the OUTPUT statement. If you wish to forecast five steps ahead, you use the option LEAD=5. The ID statement specifies the yearly interval between observations and provides the Time column in the forecast output.

The VARMAX procedure output is shown in Figure 30.2 through Figure 30.10.

```
proc varmax data=simul1;
   id date interval=year;
   model y1 y2 / p=1 noint lagmax=3;
   output lead=5;
run;
```

```
                              The VARMAX Procedure

                        Number of Observations        100
                        Number of Pairwise Missing      0


                          Simple Summary Statistics

                                                 Standard
Variable Type              N          Mean      Deviation          Min          Max

y1       Dependent       100      -0.21653        2.78210     -4.75826      8.37032
y2       Dependent       100       0.16905        2.58184     -6.04718      9.58487



                              The VARMAX Procedure

                  Type of Model                             VAR(1)
                  Estimation Method     Least Squares Estimation
```

Figure 30.2. Descriptive Statistics and Model Type

The VARMAX procedure first displays descriptive statistics. The Type column specifies that the variables are dependent variables. The column N stands for the number of nonmissing observations. The last two lines in Figure 30.2 show the type and the estimation method of the fitted model for the simulated data.

```
                              The VARMAX Procedure

                            AR Coefficient Estimates

                  Lag    Variable              y1              y2

                    1    y1               1.15977        -0.51058
                         y2               0.54634         0.38499


                                   Schematic
                                 Representation
                                  of Parameter
                                   Estimates

                                Variable/
                                Lag          AR1

                                y1           +-
                                y2           ++

                                 + is > 2*std
                                error,  - is <
                                -2*std error,
                                . is between,
                                   * is N/A


                           Model Parameter Estimates

                                            Standard
Equation Parameter      Estimate           Error t Value Pr > |t| Variable

y1       AR1_1_1         1.15977         0.05508   21.06   0.0001 y1(t-1)
         AR1_1_2        -0.51058         0.05898   -8.66   0.0001 y2(t-1)
y2       AR1_2_1         0.54634         0.05779    9.45   0.0001 y1(t-1)
         AR1_2_2         0.38499         0.06188    6.22   0.0001 y2(t-1)
```

Figure 30.3. Parameter Estimates

Figure 30.3 shows the AR coefficient matrix in terms of lag 1, the parameter estimates, and their significance, which can indicate how well the model fits the data.

The second table schematically represents the parameter estimates and allows for easy verification of their significance in matrix form.

In the last table, the first column gives the left-hand-side variable of the equation; the second column, the parameter name ARl_i_j, which indicates the (i,j)th element of the lag l autoregressive coefficient; the last column, the regressor corresponding to its parameter.

The fitted VAR(1) model with estimated standard errors in parentheses is given as

$$\mathbf{y}_t = \left(\begin{array}{rr} 1.160 & -0.511 \\ (0.055) & (0.059) \\ 0.546 & 0.385 \\ (0.058) & (0.062) \end{array} \right) \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

Clearly, all parameter estimates in the coefficient matrix Φ_1 are significant.

The model can also be written as two univariate regression equations.

$$\begin{aligned} y_{1t} &= 1.160\, y_{1,t-1} - 0.511\, y_{2,t-1} + \epsilon_{1t} \\ y_{2t} &= 0.546\, y_{1,t-1} + 0.385\, y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

```
                         The VARMAX Procedure

                      Covariances of Innovations

             Variable              y1              y2

             y1               1.28875         0.39751
             y2               0.39751         1.41839


                             Information
                               Criteria

                           AICC    0.554443
                           HQC     0.595201
                           AIC     0.552777
                           SBC      0.65763
                           FPEC    1.738092
```

Figure 30.4. Innovation Covariance Estimates and Information Criteria

The table in Figure 30.4 shows the innovation covariance matrix estimates and the various information criteria results. The smaller value of information criteria fits the data better when it is compared to other models. The variable names in the covariance matrix are printed for convenience; $y1$ means the innovation for $y1$, and $y2$ means the innovation for $y2$.

```
                         The VARMAX Procedure

                    Cross Covariances of Residuals

           Lag    Variable              y1              y2

             0    y1               1.24909         0.36398
                  y2               0.36398         1.34203
             1    y1               0.01635         0.03087
                  y2              -0.07210        -0.09834
             2    y1               0.06591         0.07835
                  y2               0.01096        -0.05780
             3    y1              -0.00203         0.08804
                  y2              -0.02129         0.07277
```

Figure 30.5. Multivariate Diagnostic Checks

```
                              The VARMAX Procedure

                        Cross Correlations of Residuals

                  Lag    Variable              y1              y2

                    0    y1               1.00000         0.28113
                         y2               0.28113         1.00000
                    1    y1               0.01309         0.02385
                         y2              -0.05569        -0.07328
                    2    y1               0.05277         0.06052
                         y2               0.00847        -0.04307
                    3    y1              -0.00163         0.06800
                         y2              -0.01644         0.05422


                       Schematic Representation of Cross
                           Correlations of Residuals
                       Variable/
                       Lag           0     1     2     3

                       y1           ++    ..    ..    ..
                       y2           ++    ..    ..    ..

                          + is > 2*std error,  - is <
                          -2*std error,  . is between
```

Figure 30.6. Multivariate Diagnostic Checks Continued

```
                              The VARMAX Procedure

                          Portmanteau Test for Cross
                           Correlations of Residuals
                 Up To
                 Lag            DF    Chi-Square    Pr > ChiSq

                       2         4          1.84        0.7659
                       3         8          2.57        0.9582
```

Figure 30.7. Multivariate Diagnostic Checks Continued

Figure 30.5, Figure 30.6, and Figure 30.7 show tests for white noise residuals. The output shows that you cannot reject the hypothesis that the residuals are uncorrelated.

```
                              The VARMAX Procedure

                      Univariate Model ANOVA Diagnostics

                                             Standard
          Variable      R-Square            Deviation    F Value    Pr > F

          y1              0.8369              1.13523     497.67    <.0001
          y2              0.7978              1.19096     382.76    <.0001
```

Figure 30.8. Univariate Diagnostic Checks

The VARMAX procedure provides diagnostic checks for the univariate form of the equations. The table in Figure 30.8 describes how well each univariate equation fits the data. From two univariate regression equations in Figure 30.3, the values of R^2 in the second column are 0.84 and 0.80 for each equation. The standard deviations in the third column are the square roots of the diagonal elements of the covariance matrix from Figure 30.4. The F statistics are in the fourth column for hypotheses to test $\phi_{11} = \phi_{12} = 0$ and $\phi_{21} = \phi_{22} = 0$, respectively, where ϕ_{ij} is the (i, j)th element of the matrix Φ_1. The last column shows the p-values of the F statistics. The results show that each univariate model is significant.

```
                                 The VARMAX Procedure

                        Univariate Model White Noise Diagnostics

                         Durbin             Normality                    ARCH
 Variable                Watson    Chi-Square     Pr > ChiSq     F Value     Pr > F

 y1                     1.96656          3.32         0.1900        0.13     0.7199
 y2                     2.13609          5.46         0.0653        2.10     0.1503


                             Univariate Model AR Diagnostics

                  AR1              AR2              AR3              AR4
Variable   F Value   Pr > F   F Value   Pr > F   F Value   Pr > F   F Value   Pr > F

y1            0.02   0.8980      0.14   0.8662      0.09   0.9629      0.82   0.5164
y2            0.52   0.4709      0.41   0.6650      0.32   0.8136      0.32   0.8664
```

Figure 30.9. Univariate Diagnostic Checks Continued

The check for white noise residuals in terms of the univariate equation is shown in Figure 30.9. This output contains information that indicates whether the residuals are correlated and heteroscedastic. In the first table, the second column contains the Durbin-Watson test statistics; the third and fourth columns show the Jarque-Bera normality test statistics and their p-values; the last two columns show F statistics and their p-values for ARCH(1) disturbances. The second table includes F statistics and their p-values for AR(1) to AR(4) disturbances.

```
                                 The VARMAX Procedure

                                      Forecasts

                                                    Standard
Variable       Obs   Time       Forecast               Error      95% Confidence Limits

y1             101   2000       -3.59212             1.13523       -5.81713       -1.36711
               102   2001       -3.09448             1.70915       -6.44435        0.25539
               103   2002       -2.17433             2.14472       -6.37792        2.02925
               104   2003       -1.11395             2.43166       -5.87992        3.65203
               105   2004       -0.14342             2.58740       -5.21463        4.92779
y2             101   2000       -2.09873             1.19096       -4.43298        0.23551
               102   2001       -2.77050             1.47666       -5.66469        0.12369
               103   2002       -2.75724             1.74212       -6.17173        0.65725
               104   2003       -2.24943             2.01925       -6.20709        1.70823
               105   2004       -1.47460             2.25169       -5.88782        2.93863
```

Figure 30.10. Forecasts

The table in Figure 30.10 gives forecasts, their prediction errors, and 95% confidence limits.

Bayesian Vector Autoregressive Process

The Bayesian Vector AutoRegressive (BVAR) model is used to avoid problems of collinearity and over-parameterization that often occur with the use of VAR models. BVAR models do this by imposing priors on the AR parameters.

The following statements fit a BVAR(1) model to the simulated data. You specify the PRIOR= option with the hyper-parameters. The options LAMBDA=0.9 and THETA=0.1 are hyper-parameters controlling the prior covariance. Part of the VARMAX procedure output is shown in Figure 30.11.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint
                 prior=(lambda=0.9 theta=0.1);
run;
```

```
                           The VARMAX Procedure

                Type of Model                            BVAR(1)
                Estimation Method    Maximum Likelihood Estimation
                Prior Lambda                                 0.9
                Prior Theta                                  0.1


                        Model Parameter Estimates

                                        Standard
  Equation Parameter       Estimate        Error t Value Pr > |t| Variable

  y1        AR1_1_1         1.05623      0.05050   20.92   0.0001 y1(t-1)
            AR1_1_2        -0.34707      0.04824   -7.19   0.0001 y2(t-1)
  y2        AR1_2_1         0.40068      0.04889    8.20   0.0001 y1(t-1)
            AR1_2_2         0.48728      0.05740    8.49   0.0001 y2(t-1)


                        Covariances of Innovations

                  Variable              y1              y2

                  y1               1.35807         0.44152
                  y2               0.44152         1.45070
```

Figure 30.11. Parameter Estimates for the BVAR(1) Model

The output in Figure 30.11 shows that parameter estimates are slightly different from those in Figure 30.3. By choosing the appropriate priors, you may be able to get more accurate forecasts using a BVAR model rather than using an unconstrained VAR model.

Vector Error Correction Model

A Vector Error Correction Model (VECM) can lead to a better understanding of the nature of any nonstationarity among the different component series and can also improve longer term forecasting over an unconstrained model.

The VECM(p) form with the cointegration rank $r(\leq k)$ is written as

$$\Delta\mathbf{y}_t = \boldsymbol{\delta} + \Pi\mathbf{y}_{t-1} + \sum_{i=1}^{p-1}\Phi_i^*\Delta\mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where Δ is the differencing operator, such that $\Delta\mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$; $\Pi = \alpha\beta'$, where α and β are $k \times r$ matrices; Φ_i^* is a $k \times k$ matrix.

It has an equivalent VAR(p) representation as described in the preceding section.

$$\mathbf{y}_t = \boldsymbol{\delta} + (I_k + \Pi + \Phi_1^*)\mathbf{y}_{t-1} + \sum_{i=2}^{p-1}(\Phi_i^* - \Phi_{i-1}^*)\mathbf{y}_{t-i} - \Phi_{p-1}^*\mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

where I_k is a $k \times k$ identity matrix.

Example of Vector Error Correction Model

An example of the second-order nonstationary vector autoregressive model is

$$\mathbf{y}_t = \begin{pmatrix} -0.2 & 0.1 \\ 0.5 & 0.2 \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix} \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

with

$$\Sigma = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix} \text{ and } \mathbf{y}_0 = 0$$

This process can be given the following VECM(2) representation with the cointegration rank one:

$$\Delta\mathbf{y}_t = \begin{pmatrix} -0.4 \\ 0.1 \end{pmatrix} (1, -2)\mathbf{y}_{t-1} - \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix} \Delta\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The following PROC IML statements generate simulated data for the VECM(2) form specified above and plot the data as shown in Figure 30.12:

```
proc iml;
   sig = 100*i(2);
   phi = {-0.2 0.1, 0.5 0.2, 0.8 0.7, -0.4 0.6};
   call varmasim(y,phi) sigma = sig n = 100 initial = 0
                        seed = 45876;
   cn = {'y1' 'y2'};
   create simul2 from y[colname=cn];
   append from y;
quit;

data simul2;
   set simul2;
   t+1;


proc gplot data=simul2;
   symbol1 v = none i = join l = 1;
   symbol2 v = none i = join l = 2;
   plot y1 * t = 1
        y2 * t = 2 / overlay;
run;
```

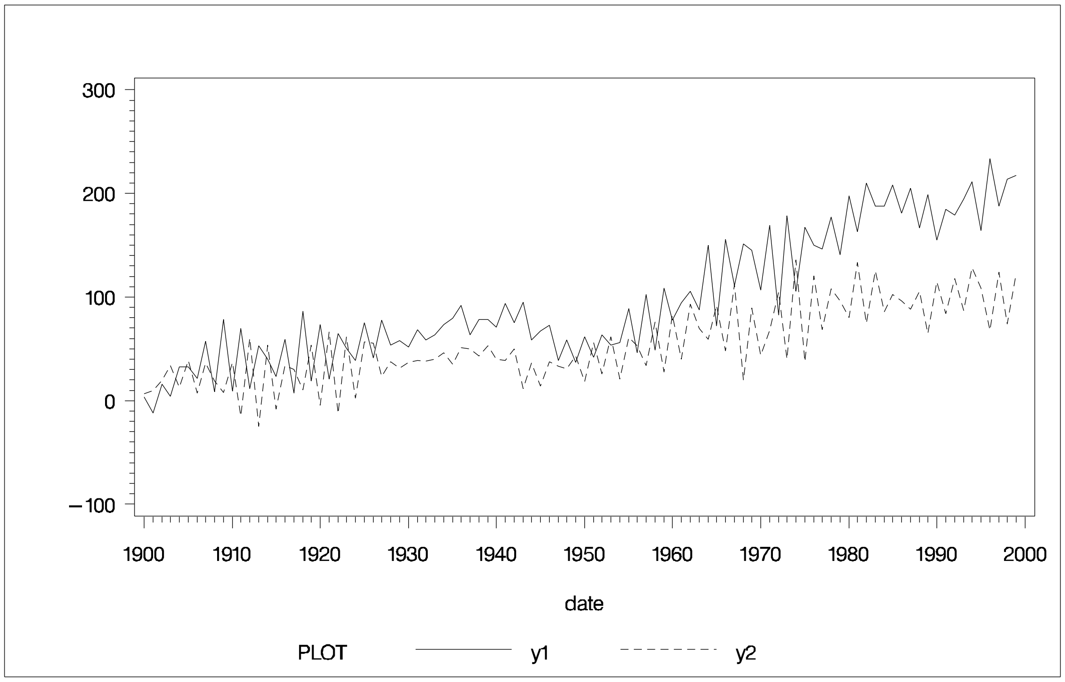


Figure 30.12. Plot of Generated Data Process

Cointegration Testing

The following statements use the Johansen cointegration rank test. The COINTTEST=(JOHANSEN) option does the Johansen trace test and is equivalent to the COINTTEST or COINTTEST=(JOHANSEN=(TYPE=TRACE)) option.

```
proc varmax data=simul2;
   model y1 y2 / p=2 noint dftest cointtest=(johansen);
run;
```

```
                              The VARMAX Procedure

                          Dickey-Fuller Unit Root Tests

   Variable     Type              Rho     Pr < Rho        Tau     Pr < Tau

   y1           Zero Mean        1.47       0.9628       1.65       0.9755
                Single Mean     -0.80       0.9016      -0.47       0.8916
                Trend          -10.88       0.3573      -2.20       0.4815
   y2           Zero Mean       -0.05       0.6692      -0.03       0.6707
                Single Mean     -6.03       0.3358      -1.72       0.4204
                Trend          -50.49       0.0003      -4.92       0.0006


                        Cointegration Rank Test Using Trace

                                                   5%
  H0:       H1:                                Critical    Drift       Drift in
 Rank=r    Rank>r    Eigenvalue        Trace      Value    in ECM      Process

      0         0        0.5086      70.7279      12.21    NOINT       Constant
      1         1        0.0111       1.0921       4.14
```

Figure 30.13. Dickey-Fuller Tests and Cointegration Rank Test

Figure 30.13 shows the output for Dickey-Fuller tests for the nonstationarity of each series and Johansen cointegration rank test between series.

In Dickey-Fuller tests, the second column specifies three types of models, which are zero mean, single mean, or trend. The third column (Rho) and the fifth column (Tau) are the test statistics for unit root testing. Other colummns are their p-values. You can see that both series have unit roots.

In the cointegration rank test, the last two columns explain the drift in the model or process. Since the NOINT option is specified, the model is

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \Phi_1^* \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The column DriftInECM means there is no separate drift in the error correction model, and the column DriftInProcess means the process has a constant drift before differencing.

H0 is the null hypothesis and H1 is the alternative hypothesis. The first row tests $r = 0$ against $r > 0$; the second row tests $r = 1$ against $r > 1$. The Trace test statistics in the fourth column are computed by $-T \sum_{i=r+1}^{k} \log(1 - \lambda_i)$ where T is the available number of observations and λ_i is the eigenvalue in the third column. By default, the critical values at 5% significance level are used for testing. You can compare the test statistics and critical values in each row; there is one cointegrated process.

The following statements fit a VECM(2) form to the simulated data. From the result in Figure 30.13, the time series are cointegrated with rank=1. You specify the ECM= option with the option RANK=1. For normalizing the value of the cointegrated vector, you specify the normalized variable with the NORMALIZE= option.

The PRINT=(IARR) option provides the VAR(2) representation. The VARMAX procedure output is shown in Figure 30.14 through Figure 30.16.

```
proc varmax data=simul2;
   model y1 y2 / p=2 noint print=(iarr) lagmax=3
                 ecm=(rank=1 normalize=y1);
run;
```

```
                          The VARMAX Procedure

             Type of Model                                  VECM(2)
             Estimation Method    Maximum Likelihood Estimation
             Cointegrated Rank                                    1


                          Long-Run Parameter Beta
                           Estimates When RANK=1

                          Variable                1

                          y1                1.00000
                          y2               -1.95575


                           Adjustment Coefficient
                              Alpha Estimates
                                When RANK=1

                          Variable                1

                          y1               -0.46680
                          y2                0.10667
```

Figure 30.14. Parameter Estimates for the VECM(2) Form

The ECM= option produces the estimates of the long-run parameter, $\boldsymbol{\beta}$, and the adjustment coefficient, $\boldsymbol{\alpha}$. In Figure 30.14, "1" indicates the first column of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ matrices. Since the cointegration rank is 1 in the bivariate system, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are two-dimensional vectors. The estimated cointegrating vector is $\hat{\boldsymbol{\beta}} = (1, -1.96)'$; the long-run relationship between y_{1t} and y_{2t} is $y_{1t} = 1.96y_{2t}$. The first element of $\hat{\boldsymbol{\beta}}$ is 1 since y_1 is specified as the normalized variable.

```
                              The VARMAX Procedure

                      Parameter Alpha * Beta' Estimates

                   Variable              y1               y2

                   y1              -0.46680          0.91295
                   y2               0.10667         -0.20862


                      AR Coefficients of Differenced Lag

              DIF Lag     Variable              y1               y2

                    1     y1              -0.74332         -0.74621
                          y2               0.40493         -0.57157


                          Model Parameter Estimates

                                             Standard
Equation Parameter       Estimate            Error t Value Pr > |t| Variable

D_y1     AR1_1_1         -0.46680         0.04786                   y1(t-1)
         AR1_1_2          0.91295         0.09359                   y2(t-1)
         AR2_1_1         -0.74332         0.04526  -16.42    0.0001 D_y1(t-1)
         AR2_1_2         -0.74621         0.04769  -15.65    0.0001 D_y2(t-1)
D_y2     AR1_2_1          0.10667         0.05146                   y1(t-1)
         AR1_2_2         -0.20862         0.10064                   y2(t-1)
         AR2_2_1          0.40493         0.04867    8.32    0.0001 D_y1(t-1)
         AR2_2_2         -0.57157         0.05128  -11.15    0.0001 D_y2(t-1)
```

Figure 30.15. Parameter Estimates for the VECM(2) Form Continued

Figure 30.15 shows the parameter estimates in terms of one lagged coefficient, $\mathbf{y}_{t-1}$, and one differenced lagged coefficient, $\Delta\mathbf{y}_{t-1}$, and their significance. “Alpha * Beta$'$” indicates the coefficient of $\mathbf{y}_{t-1}$ and is obtained by multiplying the “Alpha” and “Beta” estimates in Figure 30.14. The parameter AR1_i_j corresponds to the elements in the “Alpha * Beta$'$” matrix. The t values and p-values corresponding to the parameters AR1_i_j are missing since the parameters AR1_i_j have non-Gaussian distributions. The parameter AR2_i_j corresponds to the elements in the differenced lagged AR coefficient matrix. The “D_” prefixed to a variable name in Figure 30.15 implies differencing.

The fitted model is given as

$$\Delta\mathbf{y}_t = \left(\begin{array}{rr} -0.467 & 0.913 \\ (0.048) & (0.094) \\ 0.107 & -0.209 \\ (0.051) & (0.100) \end{array}\right)\mathbf{y}_{t-1} + \left(\begin{array}{rr} -0.743 & -0.746 \\ (0.045) & (0.048) \\ 0.405 & -0.572 \\ (0.049) & (0.051) \end{array}\right)\Delta\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

```
                          The VARMAX Procedure

                     Infinite Order AR Representation

               Lag    Variable              y1              y2

                 1    y1              -0.21013         0.16674
                      y2               0.51160         0.21980
                 2    y1               0.74332         0.74621
                      y2              -0.40493         0.57157
                 3    y1               0.00000         0.00000
                      y2               0.00000         0.00000
```

Figure 30.16. Change the VECM(2) Form to the VAR(2) Model

The PRINT=(IARR) option in the previous SAS statements prints the reparameterized coefficient estimates. For the LAGMAX=3 in the SAS statements, the coefficent matrix of lag 3 is zero.

The VECM(2) form in Figure 30.16 can be rewritten as the following second-order vector autoregressive model:

$$\mathbf{y}_t = \left(\begin{array}{rr} -0.210 & 0.167 \\ 0.512 & 0.220 \end{array} \right) \mathbf{y}_{t-1} + \left(\begin{array}{rr} 0.743 & 0.746 \\ -0.405 & 0.572 \end{array} \right) \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

Bayesian Vector Error Correction Model

Bayesian inference on a cointegrated system begins by using the priors of $\boldsymbol{\beta}$ obtained from the VECM(p) form. Bayesian vector error correction models can improve forecast accuracy for cointegrated processes. The following statements fit a BVECM(2) form to the simulated data. You specify both the PRIOR= and ECM= options for the Bayesian vector error correction model. The VARMAX procedure output is shown in Figure 30.17.

```
proc varmax data=simul2;
   model y1 y2 / p=2 noint
                 prior=(lambda=0.5 theta=0.2)
                 ecm=(rank=1 normalize=y1);
run;
```

```
                              The VARMAX Procedure

                Type of Model                             BVECM(2)
                Estimation Method    Maximum Likelihood Estimation
                Cointegrated Rank                                1
                Prior Lambda                                   0.5
                Prior Theta                                    0.2


                             Adjustment Coefficient
                                Alpha Estimates
                                  When RANK=1

                             Variable                1

                             y1               -0.34392
                             y2                0.16659


                        Parameter Alpha * Beta' Estimates

                     Variable              y1              y2

                     y1              -0.34392         0.67262
                     y2               0.16659        -0.32581


                        AR Coefficients of Differenced Lag

               DIF Lag    Variable              y1              y2

                     1    y1              -0.80070        -0.59320
                          y2               0.33417        -0.53480
```

Figure 30.17. Parameter Estimates for the BVECM(2) Form

Figure 30.17 shows the model type fitted the data, the estimates of the adjustment coefficient $\boldsymbol{\alpha}$, the parameter estimates in terms of one lagged coefficient, and one differenced lagged coefficient.

Vector Autoregressive Process with Exogenous Variables

A VAR process can be affected by other observable variables that are determined outside the system of interest. Such variables are called exogenous (independent) variables. Exogenous variables can be stochastic or nonstochastic. The process can also be affected by the lags of exogenous variables. A model used to describe this process is called a VARX(p,s) model.

The VARX(p,s) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\mathbf{x}_t = (x_{1t}, \ldots, x_{rt})'$ is an r-dimensional time series vector and Θ_i^* is a $k \times r$ matrix.

For example, a VARX(1,0) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \Theta_0^* \mathbf{x}_t + \boldsymbol{\epsilon}_t$$

where $\mathbf{y}_t = (y_{1t}, y_{2t}, y_{3t})'$ and $\mathbf{x}_t = (x_{1t}, x_{2t})'$.

The following statements fit the VARX(1,0) model to the given data:

```
data grunfeld;
   input year y1 y2 y3 x1 x2 x3;
   label y1='Gross Investment GE'
         y2='Capital Stock Lagged GE'
         y3='Value of Outstanding Shares GE Lagged'
         x1='Gross Investment W'
         x2='Capital Stock Lagged W'
         x3='Value of Outstanding Shares Lagged W';
   datalines;
         1935  33.1 1170.6  97.8 12.93  191.5   1.8
         1936  45.0 2015.8 104.4 25.90  516.0    .8
         1937  77.2 2803.3 118.0 35.05  729.0   7.4
         1938  44.6 2039.7 156.2 22.89  560.4  18.1
         1939  48.1 2256.2 172.6 18.84  519.9  23.5
         1940  74.4 2132.2 186.6 28.57  628.5  26.5
         1941 113.0 1834.1 220.9 48.51  537.1  36.2
         1942  91.9 1588.0 287.8 43.34  561.2  60.8
         1943  61.3 1749.4 319.9 37.02  617.2  84.4
         1944  56.8 1687.2 321.3 37.81  626.7  91.2
         1945  93.6 2007.7 319.6 39.27  737.2  92.4
         1946 159.9 2208.3 346.0 53.46  760.5  86.0
         1947 147.2 1656.7 456.4 55.56  581.4 111.1
         1948 146.3 1604.4 543.4 49.56  662.3 130.6
         1949  98.3 1431.8 618.3 32.04  583.8 141.8
         1950  93.5 1610.5 647.4 32.24  635.2 136.7
         1951 135.2 1819.4 671.3 54.38  723.8 129.7
         1952 157.3 2079.7 726.1 71.78  864.1 145.5
         1953 179.5 2371.6 800.3 90.08 1193.5 174.8
         1954 189.6 2759.9 888.9 68.60 1188.9 213.5
         ;

proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1;
run;
```

The VARMAX procedure output is shown in Figure 30.18 through Figure 30.20.

```
                              The VARMAX Procedure

                        Number of Observations        20
                        Number of Pairwise Missing     0


                           Simple Summary Statistics

                                                   Standard
Variable Type              N          Mean        Deviation           Min           Max

y1       Dependent        20     102.29000         48.58450      33.10000     189.60000
y2       Dependent        20    1941.32500        413.84329    1170.60000    2803.30000
y3       Dependent        20     400.16000        250.61885      97.80000     888.90000
x1       Independent      20      42.89150         19.11019      12.93000      90.08000
x2       Independent      20     670.91000        222.39193     191.50000    1193.50000

                           Simple Summary Statistics

                 Variable Label

                 y1       Gross Investment GE
                 y2       Capital Stock Lagged GE
                 y3       Value of Outstanding Shares GE Lagged
                 x1       Gross Investment W
                 x2       Capital Stock Lagged W
```

Figure 30.18. Descriptive Statistics for the VARX(1, 0) Model

Figure 30.18 shows the descriptive statistics for the dependent (endogenous) and independent (exogenous) variables with labels.

```
                          The VARMAX Procedure

             Type of Model                         VARX(1,0)
             Estimation Method    Least Squares Estimation


                            Constant Estimates

                        Variable      Constant

                        y1             -12.01279
                        y2             702.08673
                        y3             -22.42110


             Coefficient Estimates of Independent Variables

              Lag    Variable              x1              x2

                0    y1               1.69281        -0.00859
                     y2              -6.09850         2.57980
                     y3              -0.02317        -0.01274


                        AR Coefficient Estimates

       Lag    Variable              y1              y2              y3

         1    y1               0.23699         0.00763         0.02941
              y2              -2.46656         0.16379        -0.84090
              y3               0.95116         0.00224         0.93801


                        Schematic Representation
                         of Parameter Estimates

                      Variable/
                      Lag          C     XL0    AR1

                      y1           .     +.     ...
                      y2           +     .+     ...
                      y3           -     ..     +.+

                         + is > 2*std error,  -
                         is < -2*std error,  .
                         is between,  * is N/A
```

Figure 30.19. Parameter Estimates for the VARX(1, 0) Model

Figure 30.19 shows the parameter estimates for the constant, the lag zero coefficients of exogenous variables, and the lag one AR coefficients. From the schematic representation of parameter estimates, the significance of the parameter estimates can be easily verified. The symbol “C” means the constant and “XL0” the lag zero coefficients of exogenous variables.

```
                              The VARMAX Procedure

                           Model Parameter Estimates

                                             Standard
Equation Parameter       Estimate        Error t Value Pr > |t| Variable

y1       CONST1        -12.01279     27.47108   -0.44   0.6691 1
         XL0_1_1         1.69281      0.54395    3.11   0.0083 x1(t)
         XL0_1_2        -0.00859      0.05361   -0.16   0.8752 x2(t)
         AR1_1_1         0.23699      0.20668    1.15   0.2722 y1(t-1)
         AR1_1_2         0.00763      0.01627    0.47   0.6470 y2(t-1)
         AR1_1_3         0.02941      0.04852    0.61   0.5548 y3(t-1)
y2       CONST2        702.08673    256.48046    2.74   0.0169 1
         XL0_2_1        -6.09850      5.07849   -1.20   0.2512 x1(t)
         XL0_2_2         2.57980      0.50056    5.15   0.0002 x2(t)
         AR1_2_1        -2.46656      1.92967   -1.28   0.2235 y1(t-1)
         AR1_2_2         0.16379      0.15193    1.08   0.3006 y2(t-1)
         AR1_2_3        -0.84090      0.45304   -1.86   0.0862 y3(t-1)
y3       CONST3        -22.42110     10.31166   -2.17   0.0487 1
         XL0_3_1        -0.02317      0.20418   -0.11   0.9114 x1(t)
         XL0_3_2        -0.01274      0.02012   -0.63   0.5377 x2(t)
         AR1_3_1         0.95116      0.07758   12.26   0.0001 y1(t-1)
         AR1_3_2         0.00224      0.00611    0.37   0.7201 y2(t-1)
         AR1_3_3         0.93801      0.01821   51.50   0.0001 y3(t-1)
```

Figure 30.20. Parameter Estimates for the VARX(1, 0) Model Continued

Figure 30.20 shows the parameter estimates and their significance.

The fitted model is given as

$$\begin{pmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \end{pmatrix} = \begin{pmatrix} -12.013 \\ (27.471) \\ 702.086 \\ (256.480) \\ -22.421 \\ (10.312) \end{pmatrix} + \begin{pmatrix} 1.693 & -0.009 \\ (0.544) & (0.054) \\ -6.099 & 2.580 \\ (5.078) & (0.501) \\ -0.023 & -0.013 \\ (0.204) & (0.020) \end{pmatrix} \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix}$$

$$+ \begin{pmatrix} 0.237 & 0.008 & 0.029 \\ (0.207) & (0.016) & (0.049) \\ -2.467 & 0.164 & -0.841 \\ (1.930) & (0.152) & (0.453) \\ 0.951 & 0.002 & 0.938 \\ (0.078) & (0.006) & (0.018) \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \end{pmatrix}$$

Parameter Estimation and Testing on Restrictions

In the previous example, the VARX(1,0) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Theta_0^* \mathbf{x}_t + \Phi_1 \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

with

$$\Theta_0^* = \begin{pmatrix} \theta_{11}^* & \theta_{12}^* \\ \theta_{21}^* & \theta_{22}^* \\ \theta_{31}^* & \theta_{32}^* \end{pmatrix} \quad \Phi_1 = \begin{pmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{pmatrix}$$

In Figure 30.20, you can see the coefficients XL_0_1_2, AR_1_1_2, and AR_1_3_2 are insignificant. The following statements restrict the coefficients of $\theta_{12}^* = \phi_{12} = \phi_{32} = 0$ for the VARX(1,0) model.

```
proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1;
   restrict XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

```
                        The VARMAX Procedure

             Coefficient Estimates of Independent Variables

              Lag    Variable             x1            x2

                0    y1              1.67592       0.00000
                     y2             -6.30880       2.65308
                     y3             -0.03576      -0.00919


                      AR Coefficient Estimates

      Lag    Variable             y1            y2            y3

        1    y1              0.27671       0.00000       0.01747
             y2             -2.16968       0.10945      -0.93053
             y3              0.96398       0.00000       0.93412


                      Schematic Representation
                       of Parameter Estimates

                     Variable/
                     Lag          C    XL0    AR1

                     y1           .    +*     .*.
                     y2           +    .+     ..-
                     y3           -    ..     +*+

                        + is > 2*std error,  -
                        is < -2*std error,  .
                        is between,  * is N/A
```

Figure 30.21. Parameter Estimation on Restrictions

The output in Figure 30.21 shows that three parameters θ_{12}^*, ϕ_{12}, and ϕ_{32} are replaced by the restricted values, zeros. From the schematic representation of parameter estimates, the three parameters θ_{12}^*, ϕ_{12}, and ϕ_{32} are replaced by $*$.

```
                              The VARMAX Procedure

                      Testing of the Restricted Parameters

                                              Standard
        Parameter          Estimate              Error     t Value     Pr > |t|

        XL0_1_2             1.74969           21.44026        0.08       0.9389
        AR1_1_2            30.36254           70.74347        0.43       0.6899
        AR1_3_2            55.42191          164.03075        0.34       0.7524
```

Figure 30.22. RESTRICT Statement Results

The output in Figure 30.22 shows the estimates of the Lagrangian parameters and their significance. You cannot reject the null hypotheses $\theta^*_{12} = 0$, $\phi_{12} = 0$, and $\phi_{32} = 0$ with the 0.05 significance level.

The TEST statement in the following example tests $\phi_{31} = 0$ and $\theta^*_{12} = \phi_{12} = \phi_{32} = 0$ for the VARX(1,0) model:

```
proc varmax data=grunfeld;
   model y1-y3 = x1 x2/ p=1;
   test AR(1,3,1)=0;
   test XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

```
                              The VARMAX Procedure

                           Testing of the Parameters

                      Test         DF     Chi-Square     Pr > ChiSq

                         1          1         150.31         <.0001
                         2          3           0.34         0.9522
```

Figure 30.23. TEST Statement Results

The output in Figure 30.23 shows that the first column in the output is the index corresponding to each TEST statement; you can reject the hypothesis test $\phi_{31} = 0$ at the 0.05 significance level; you cannot reject the joint hypothesis test $\theta^*_{12} = \phi_{12} = \phi_{32} = 0$ at the 0.05 significance level.

Causality Testing

The following statements use the CAUSAL statement to compute the Granger-Causality test for the VAR(1) model. For the Granger-Causality tests, the autoregressive order should be defined by the option P= in the MODEL statement. The variables in groups are defined in the MODEL statement as well. Regardless whether the variables in groups are either dependent or independent in the MODEL statement, the CAUSAL statement fits the VAR(p) model using the variables in two groups considering them as dependent variables.

```
proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1 noprint;
   causal group1=(x1) group2=(y1-y3);
   causal group1=(y3) group2=(y1 y2);
run;
```

```
                         The VARMAX Procedure

                      Granger-Causality Wald Test

                Test        DF    Chi-Square    Pr > ChiSq

                   1         3          2.40        0.4946
                   2         2        262.88        <.0001


                 Test 1:  Group 1 Variables:  x1
                          Group 2 Variables:  y1 y2 y3


                   Test 2:  Group 1 Variables:  y3
                            Group 2 Variables:  y1 y2
```

Figure 30.24. CAUSAL Statement Results

The output in Figure 30.24 is associated with the CAUSAL statement. The first CAUSAL statement fits the VAR(1) model using the variabls $y1$, $y2$, $y3$, and $x1$; the second CAUSAL statement fits the VAR(1) model using the variabls $y1$, $y3$, and $y2$.

The first column in the output is the index corresponding to each CAUSAL statement. The output shows that you cannot reject that $x1$ is influenced by itself and not by $(y1, y2, y3)$ at the 0.05 significance level for Test 1; you can reject that $y3$ is influenced by itself and not by $(y1, y2)$ for Test 2.

Syntax

PROC VARMAX *options* **;**
  BY *variables* **;**
  CAUSAL group1 = *(variables)* **group2** = *(variables)* **;**
  COINTEG rank = *number* < *options* > **;**
  ID *variable* **interval**= *value* < *option* > **;**
  MODEL *dependent variables* < = *regressors* >
    <, *dependent variables* < = *regressors* > ... >
    </ *options* > **;**
  NLOPTIONS *options* **;**
  OUTPUT < *options* > **;**
  RESTRICT *restrictions* **;**
  TEST *restrictions* **;**

Functional Summary

The statements and options used with the VARMAX procedure are summarized in the following table:

Description	Statement	Option
Data Set Options		
specify the input data set	VARMAX	DATA=
write parameter estimates to an output data set	VARMAX	OUTEST=
include covariances in the OUTEST= data set	VARMAX	OUTCOV
write the diagnostic checking tests for a model and the cointegration test results to an output data set	VARMAX	OUTSTAT=
write actuals, predictions, residuals, and confidence limits to an output data set	OUTPUT	OUT=
BY Groups		
specify BY-group processing	BY	
ID Variable		
specify identifying variable	ID	
specify the time interval between observations	ID	INTERVAL=
control the alignment of SAS Date values	ID	ALIGN=
Options to Control the Optimization Process		
specify the optimization options	NLOPTIONS	see Chapter 10
Printing Control Options		
specify how many lags to print results	MODEL	LAGMAX=
suppress the printed output	MODEL	NOPRINT
request all printing options	MODEL	PRINTALL
request the printing format	MODEL	PRINTFORM=
PRINT= Option		
print the correlation matrix of parameter estimates	MODEL	CORRB
print the cross-correlation matrices of independent variables	MODEL	CORRX
print the cross-correlation of dependent variables	MODEL	CORRY
print the covariance matrices of prediction errors	MODEL	COVPE
print the cross-covariance matrices of the independent variables	MODEL	COVX
print the cross-covariance matrices of the dependent variables	MODEL	COVY
print the covariance matrix of parameter estimates	MODEL	COVB
print the decomposition of the prediction error covariance matrix	MODEL	DECOMPOSE

Description	Statement	Option
print the contemporaneous relationships among the components of the vector time series	MODEL	DYNAMIC
print the infinite order AR representation	MODEL	IARR
print the impulse response function	MODEL	IMPULSE=
print the impulse response function in the transfer function	MODEL	IMPULSX=
print the partial autoregressive coefficient matrices	MODEL	PARCOEF
print the partial canonical correlation matrices	MODEL	PCANCORR
print the partial correlation matrices	MODEL	PCORR
print the eigenvalues of the companion matrix	MODEL	ROOTS
print the Yule-Walker estimates	MODEL	YW
Model Estimation and Order Selection Options		
center the dependent variables	MODEL	CENTER
specify the degrees of differencing for the specified model variables	MODEL	DIF=
specify the degrees of differencing for all independent variables	MODEL	DIFX=
specify the degrees of differencing for all dependent variables	MODEL	DIFY=
specify the vector error correction model	MODEL	ECM=
specify the GARCH-type model	MODEL	GARCH=
specify the estimation method	MODEL	METHOD=
select the tentative order	MODEL	MINIC=
suppress the current values of independent variables	MODEL	NOCURRENTX
suppress the intercept parameters	MODEL	NOINT
specify the number of seasonal periods	MODEL	NSEASON=
specify the order of autoregressive polynomial	MODEL	P=
specify the Bayesian prior model	MODEL	PRIOR=
specify the order of moving-average polynomial	MODEL	Q=
center the seasonal dummies	MODEL	SCENTER
specify the degree of time trend polynomial	MODEL	TREND=
specify the denominator for error covariance matrix estimates	MODEL	VARDEF=
specify the lag order of independent variables	MODEL	XLAG=
Cointegration Related Options		
print the results from the weak exogeneity test of the long-run parameters	COINTEG	EXOGENEITY
specify the restriction on the cointegrated coefficient matrix	COINTEG	H=
specify the restriction on the adjustment coefficient matrix	COINTEG	J=

Description	Statement	Option
specify the variable name whose cointegrating vectors are normalized	COINTEG	NORMALIZE=
specify a cointegration rank	COINTEG	RANK=
print the Johansen cointegration rank test	MODEL	COINTTEST= (JOHANSEN=)
print the Stock-Watson common trends test	MODEL	COINTTEST=(SW=)
print the Dickey-Fuller unit root test	MODEL	DFTEST=
Tests and Restrictions on Parameters		
test the Granger-Causality	CAUSAL	GROUP1= GROUP2=
place and test restrictions on parameter estimates	RESTRICT	
test hypotheses	TEST	
Output Control Options		
specify the size of confidence limits for forecasting	OUTPUT	ALPHA=
start forecasting before end of the input data	OUTPUT	BACK=
specify how many periods to forecast	OUTPUT	LEAD=
suppress the printed forecasts	OUTPUT	NOPRINT

PROC VARMAX Statement

PROC VARMAX *options* **;**

The following options can be used in the PROC VARMAX statement:

DATA= *SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, the PROC VARMAX statement uses the most recently created SAS data set.

OUTEST= *SAS-data-set*
writes the parameter estimates to the output data set.

COVOUT
OUTCOV
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

OUTSTAT= *SAS-data-set*
writes residual diagnostic results to an output data set. If the COINTTEST=(JOHANSEN) option is specified, the results of this option are also written to the output data set.

The following statements are the examples of the options in the PROC VARMAX statement:

```
proc varmax data=one outest=est outcov outstat=stat;
   model y1-y3 / p=1;
run;
```

```
proc varmax data=one outest=est outstat=stat;
   model y1-y3 / p=1 cointtest=(johansen);
run;
```

Other Options

In addition, any of the following MODEL statement options can be specified in the PROC VARMAX statement, which is equivalent to specifying the option for every MODEL statement: CENTER, DFTEST=, DIF=, DIFX=, DIFY=, LAGMAX=, METHOD=, MINIC=, NOCURRENTX, NOINT, NOPRINT, NSEASON=, P=, PRINT=, PRINTALL, PRINTFORM=, Q=, SCENTER, TREND=, VARDEF=, and XLAG= options.

The following statement is an example of the options in the PROC VARMAX statement:

```
proc varmax data=one lagmax=3 method=ml;
   model y1-y3 / p=1;
run;
```

BY Statement

BY *variables***;**

A BY statement can be used with the PROC VARMAX statement to obtain separate analyses on observations in groups defined by the BY variables.

The following statement is an example of the BY statement:

```
proc varmax data=one;
   by region;
   model y1-y3 / p=1;
run;
```

CAUSAL Statement

CAUSAL GROUP1=*(variables)* **GROUP2=***(variables)***;**

A CAUSAL statement prints the Granger-Causality test by fitting the VAR(p) model using all variables defined in GROUP1 and GROUP2. Any number of CAUSAL statements can be specified. The CAUSAL statement proceeds with the MODEL statement and uses the variables and the autoregressive order, p, specified in the MODEL statement. Variables in the GROUP1= and GROUP2= options should be defined in the MODEL statement. If the option P=0 is specified in the MODEL statement, the CAUSAL statement is not applicable.

The null hypothesis of the Granger-Causality test is that GROUP1 is influenced only by itself, and not by GROUP2. If the test of hypothesis fails to reject the null, the variables in the GROUP1 may be considered as independent variables.

See the "VAR Modeling" section on page 1766 for details.

The following is an example of the CAUSAL statement. You specify the CAUSAL statement with the GROUP1= and GROUP2= options.

```
proc varmax data=one;
   model y1-y3 = x1 / p=1;
   causal group1=(x1) group2=(y1-y3);
   causal group1=(y2) group2=(y1 y3);
run;
```

The first CAUSAL statement fits the VAR(1) model using the variables $y1$, $y2$, $y3$, and $x1$ and tests the null hypothesis that $x1$ causes the other variables $y1$, $y2$, and $y3$, but the other variables do not cause $x1$. The second CAUSAL statement fits the VAR(1) model using the variables $y1$, $y3$, and $y2$ and tests the null hypothesis that $y2$ causes the other variables $y1$ and $y3$, but the other variables do not cause $y2$.

COINTEG Statement

COINTEG RANK= *number* < **H=** *(matrix)* > < **J=** *(matrix)* >
< **EXOGENEITY** > < **NORMALIZE=** *variable* > **;**

The COINTEG statement fits the vector error correction model to the data, tests the restrictions of the long-run parameters and the adjustment parameters, and tests for the weak exogeneity in the long-run parameters. The cointegrated system uses the maximum likelihood analysis proposed by Johansen and Juselius (1990) and Johansen (1995a, 1995b). Only one COINTEG statement is allowed.

You specify the ECM= option or the COINTEG statement for fitting the VECM(p) with the P= option in the MODEL statement.

The following statements are equivalent for fitting the VECM(2).

```
proc varmax data=one;
   model y1-y3 / p=2 ecm=(rank=1);
run;
```

```
proc varmax data=one;
   model y1-y3 / p=2;
   cointeg rank=1;
run;
```

For testing of the restrictions of either α or β or both, you specify either J= or H= or both. You specify the EXOGENEITY option for tests of the weak exogeneity in the long-run parameters.

The following statement is an example of the COINTEG statement.

```
proc varmax data=one;
   model y1-y3 / p=2;
   cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1)
           j=(1 0, 0 0, 0 1) exogeneity;
run;
```

The following options can be used in the COINTEG statement:

EXOGENEITY

formulates the likelihood ratio tests for testing weak exogeneity in the long-run parameters. The hypothesis is that one variable is weakly exogenous for the others.

H= (*matrix*)

specifies the restrictions H on the $k \times r$ or $(k+1) \times r$ cointegrated coefficient matrix $\boldsymbol{\beta}$ such that $\boldsymbol{\beta} = H\phi$, where H is known and ϕ is unknown. The $k \times m$ or $(k+1) \times m$ matrix H is specified using this option, where k is the number of dependent variables, and m is $r \leq m < k$ with the option RANK=r. For example, consider that the system contains four variables and the option RANK=1 with $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4)'$. The restriction matrix for the test of $\beta_1 + \beta_2 = 0$ can be specified as

```
cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1);
```

Here the matrix H is 4×3 where $k = 4$ and $m = 3$, and each row of the matrix H is divided by commas (,).

When the series has no separate deterministic trend, the constant term should be restricted by $\boldsymbol{\alpha}'_{\perp}\boldsymbol{\delta} = 0$. In the example above, the $\boldsymbol{\beta}$ can be either $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4, 1)'$ or $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3, \beta_4, t)'$. Then, you can specify the restriction matrix for the previous test of $\beta_1 + \beta_2 = 0$ as follows:

```
cointeg rank=1
   h=(1 0 0 0, -1 0 0 0, 0 1 0 0, 0 0 1 0, 0 0 0 1);
```

When the cointegrated system contains three dependent variables and the option RANK=2, you can specify the restriction matrix for the test of $\beta_{1j} = -\beta_{2j}$ for $j = 1, 2$ as follows:

```
cointeg rank=2 h=(1 0, -1 0, 0 1);
```

J= (*matrix*)

specifies the restrictions J on the $k \times r$ adjustment matrix $\boldsymbol{\alpha}$ such that $\boldsymbol{\alpha} = J\psi$, where J is known and ψ is unknown. The $k \times m$ matrix J is specified using this option, where k is the number of dependent variables, and m is $r \leq m < k$ with the option RANK=r. For example, when the system contains four variables and the option RANK=1, you can specify the restriction matrix for the test of $\alpha_j = 0$ for $j = 2, 3, 4$ as follows:

```
cointeg rank=1 j=(1, 0, 0, 0);
```

When the system contains three variables and the option RANK=2, you can specify the restriction matrix for the test of $\alpha_{2j} = 0$ for $j = 1, 2$ as follows:

```
cointeg rank=2 j=(1 0, 0 0, 0 1);
```

NORMALIZE= *variable*
specifies a single dependent (endogenous) variable name whose cointegrating vectors are normalized. If the variable name is different from that specified in the COINTTEST=(JOHANSEN=) or ECM= option in the MODEL statement, the variable name specified in the COINTEG statement is used. If the normalized variable is not specified, cointegrating vectors are not normalized.

RANK= *number*
specifies the cointegration rank of the cointegrated system. This option is required in the COINTEG statement. The rank of cointegration should be greater than zero and less than the number of dependent (endogenous) variables. If the value of the RANK= option in the COINTEG statement is different from that specified in the ECM= option, the rank specified in the COINTEG statement is used.

ID Statement

ID *variable* **INTERVAL=** *value* < **ALIGN=** *value* > **;**

The ID statement specifies a variable that identifies observations in the input data set. The variable specified in the ID statement is included in the OUT= data set. Note that the ID *variable* is usually a SAS date valued variable. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= option.

ALIGN= *value*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING. The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

INTERVAL= *value*
specifies the time interval between observations. This option is required in the ID statement. The option INTERVAL=*value* is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data.

The following statement is an example of the ID statement:

```
proc varmax data=one;
   id date interval=qtr align=mid;
   model y1-y3 / p=1;
run;
```

MODEL Statement

MODEL *dependents < = regressors >*
<, dependents < = regressors > ... >
</ options > **;**

The MODEL statement specifies dependent (endogenous) variables and independent (exogenous) variables for the VARMAX model. The multivariate model can have the same or different independent variables corresponding to the dependent variables. As a special case, the VARMAX procedure allows you to analyze one dependent variable. The one MODEL statement is required.

For example, the following statements are equivalent ways of specifying the multivariate model for the vector $(y1, y2, y3)$:

```
model y1 y2 y3 </options>;
model y1-y3 </options>;
```

The following statements are equivalent ways of specifying the multivariate model for the vectors $(y1, y2, y3, y4)$ and $(x1, x2, x3, x4, x5)$:

```
model y1 y2 y3 y4 = x1 x2 x3 x4 x5 </options>;
model y1 y2 y3 y4 = x1-x5 </options>;
model y1 = x1-x5, y2 = x1-x5, y3 y4 = x1-x5 </options>;
model y1-y4 = x1-x5 </options>;
```

When the multivariate model has different independent variables corresponding to the dependent variables, equations are separated by commas (,) and the model can be specified as illustrated by the following MODEL statement:

```
model y1 = x1-x3, y2 = x3-x5, y3 y4 = x1-x5 </options>;
```

The following options can be used in the MODEL statement after a forward slash (/):

General Options

CENTER
centers dependent (endogenous) variables by subtracting their means. Note that centering is done after differencing when the DIF= or DIFY= option is specified. If there are exogenous (independent) variables, this option is not applicable.

```
model y1 y2 / p=1 center;
```

DIF(*variable***(***number-list***)<...** *variable***(***number-list***)>)**
DIF= (*variable***(***number-list***)<...** *variable***(***number-list***)>)**
specifies the degrees of differencing to be applied to the specified dependent or independent variables. The differencing can be the same for all variables, or it can vary

among variables. For example, the option DIF=(y_1(1,4) y_3(1) x_2(2)) specifies that the series y_1 is differenced at lag 1 and at lag 4, which is

$$(1 - B^4)(1 - B)y_{1t} = (y_{1t} - y_{1,t-1}) - (y_{1,t-4} - y_{1,t-5})$$

the series y_3 is differenced at lag 1, which is $(y_{3t} - y_{3,t-1})$; the series x_2 is differenced at lag 2, which is $(x_{2t} - x_{2,t-2})$.

```
   model y1 y2 = x1 x2 / p=1 dif=(y1(1) x2(2));
```

When you fit the model above, you use the data $dy1$, $y2$, $x1$, and $dx2$, where $dy1 = (1 - B)y_{1t}$ and $dx2 = (1 - B)x_{2t}$.

DIFX(*number-list***)**
DIFX= (*number-list***)**
specifies the degrees of differencing to be applied to all exogenous (independent) variables. For example, the option DIFX=(1) specifies that the series are differenced once at lag 1; the option DIFX=(1,4) specifies that the series are differenced at lag 1 and at lag 4. If exogenous variables are specified in the DIF= option, this option is ignored.

```
   model y1 y2 = x1 x2 / p=1 difx(1);
```

When you fit the model above, you use the data $y1$, $y2$, $dx1$, and $dx2$, where $dx1 = (1 - B)x_{1t}$ and $dx2 = (1 - B)x_{2t}$.

DIFY(*number-list***)**
DIFY= (*number-list***)**
specifies the degrees of differencing to be applied to all dependent (endogenous) variables. For details, see the DIFX= option. If dependent variables are specified in the DIF= option, this option is ignored.

```
   model y1 y2 / p=1 dify(1);
```

METHOD= *value*
requests the type of estimates to be computed. The possible values of the METHOD= option are

LS	specifies least-squares estimates.
ML	specifies maximum likelihood estimates.

When the options ECM=, GARCH=, PRIOR=, and Q= are specified, the default MLE method is used regardless of the method given by the option METHOD=. The option XLAG= is specified, the default LS method is used regardless of the method given by the option METHOD=. The pure VAR model uses the method given by the option METHOD=, but the default for the pure VAR model is the LS method.

```
   model y1 y2 / p=1 method=ml;
```

NOCURRENTX

suppresses the current values x_t of exogenous (independent) variables. In general, the VARX(p, s) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

For this model with $p = 1$ and $s = 2$,

```
   model y1 y2 = x1 x2 / p=1 xlag=2;
```

If this option is specified, it suppresses the current values $\mathbf{x}_t$ and starts with $\mathbf{x}_{t-1}$.

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=1}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

For this model with $p = 1$ and $s = 2$,

```
   model y1 y2 = x1 x2 / p=1 xlag=2 nocurrentx;
```

NOINT

suppresses the intercept parameter $\boldsymbol{\delta}$.

```
   model y1 y2 / p=1 noint;
```

NSEASON= *number*

specifies the number of seasonal periods. When the NSEASON=*number* option is specified, (*number*-1) seasonal dummies are added to the regressors. If the NOINT option is specified, the NSEASON= option is not applicable.

```
   model y1 y2 / p=1 nseason=4;
```

SCENTER

centers seasonal dummies specified by the NSEASON= option. The centered seasonal dummies are generated by $c - (1/s)$, where c is a seasonal dummy generated by the NSEASON=s option.

```
   model y1 y2 / p=1 nseason=4 scenter;
```

TREND= *value*

specifies the degree of deterministic time trend included in the model. Valid values are as follows:

LINEAR	includes a linear time trend as a regressor.
QUAD	includes linear and quadratic time trends as regressors.

The TREND=QUAD option is not applicable for a cointegration analysis.

```
model y1 y2 / p=1 trend=linear;
```

VARDEF= *value*

corrects for the degrees of freedom of the denominator. This option is used to calculate an error covariance matrix for the option METHOD=LS. If the option METHOD=ML is specified, the option VARDEF=N is used. Valid values are as follows:

DF	specifies that the number of nonmissing observation minus the number of regressors be used.
N	specifies that the number of nonmissing observation be used.

```
model y1 y2 / p=1 vardef=n;
```

Printing Control Options

LAGMAX= *number*

specifies the lag to compute and display the results obtained by the option PRINT=(CORRX CORRY COVX COVY IARR IMPULSE= IMPULSX= PARCOEF PCANCORR PCORR). This option is also used to print cross-covariances and cross-correlations of residuals. The default is LAGMAX=min(12, T-2), where T is the number of nonmissing observations.

```
model y1 y2 / p=1 lagmax=6;
```

NOPRINT

suppresses all printed output.

```
model y1 y2 / p=1 noprint;
```

PRINTALL

requests all printing control options. The options set by the option PRINTALL are DFTEST=, MINIC=, PRINTFORM=BOTH, and PRINT=(CORRB CORRX CORRY COVB COVPE COVX COVY DECOMPOSE DYNAMIC IARR IMPULSE=(ALL) IMPULSX=(ALL) PARCOEF PCANCORR PCORR ROOTS YW).

You can also specify this option as the option ALL.

```
   model y1 y2 / p=1 printall;
```

PRINTFORM= *value*

requests the printing format of outputs of the PRINT= option and cross-covariances and cross-correlations of residuals. Valid values are as follows:

BOTH	prints outputs in both MATRIX and UNIVARIATE forms.
MATRIX	prints outputs in matrix form. This is the default.
UNIVARIATE	prints outputs by variables.

```
   model y1 y2 / p=1 print=(impulse) printform=univariate;
```

Printing Options

PRINT=(*options***)**

The following options can be used in the PRINT=( ) option. The options are listed within parentheses.

CORRB

prints the estimated correlations of the parameter estimates. If the number in parentheses follow the options listed below, the options print the number of lags specified by *number* in parentheses. The default is the number of lags specified by the option LAGMAX=*number*.

CORRX
CORRX(*number***)**

prints the cross-correlation matrices of exogenous (independent) variables.

CORRY
CORRY(*number***)**

prints the cross-correlation matrices of dependent (endogenous) variables.

COVB

prints the estimated covariances of the parameter estimates.

COVPE
COVPE(*number***)**

prints the covariance matrices of *number*-ahead prediction errors for the VARMAX(p,q,s) model. If the DIF= or DIFY= option is specified, the covariance matrices of multistep-ahead prediction errors are computed based on the differenced data. This option is not applicable when the PRIOR= option is specified. See the “Forecasting” section on page 1756 for details.

COVX
COVX(*number***)**

prints the cross-covariance matrices of exogenous (independent) variables.

COVY
COVY(*number*)

prints the cross-covariance matrices of dependent (endogenous) variables.

DECOMPOSE
DECOMPOSE(*number*)

prints the decomposition of the prediction error covariances using the number of lags specified by *number* in parentheses for the VARMA(p,q) model. It can be interpreted as the contribution of innovations in one variable to the mean squared error of the multistep-ahead forecast of another variable. The DECOMPOSE option also prints proportions of the forecast error variance.

If the DIF= or DIFY= option is specified, the covariance matrices of multistep-ahead prediction errors are computed based on the differenced data. This option is not applicable when the PRIOR= option is specified. See the "Forecasting" section on page 1756 for details.

DYNAMIC

prints the contemporaneous relationships among the components of the vector time series.

IARR
IARR(*number*)

prints the infinite order AR representation of a VARMA process. If the ECM= option is specified, the reparameterized AR coefficient matrices are printed.

IMPULSE
IMPULSE(*number*)
IMPULSE= (SIMPLE ACCUM ORTH STDERR ALL)
IMPULSE(*number*)= (SIMPLE ACCUM ORTH STDERR ALL)

prints the impulse response function. It investigates the response of one variable to an impulse in another variable in a system that involves a number of other variables as well. It is an infinite order MA representation of a VARMA process. See the "Forecasting" section on page 1756 for details.

The following options can be used in the IMPULSE=( ) option. The options are listed within parentheses.

ACCUM	prints the accumulated impulse function.
ALL	equivalent to specifying all of SIMPLE, ACCUM, ORTH, and STDERR.
ORTH	prints the orthogonalized impulse function.
SIMPLE	prints the impulse response function. This is the default.
STDERR	prints the standard errors of the impulse response function, the accumulated impulse response function, or the orthogonalized impulse response function. If the exogenous variables are used to fit the model, this option is ignored.

IMPULSX
IMPULSX(*number*)
IMPULSX= (SIMPLE ACCUM ALL)

IMPULSX(*number*)= (SIMPLE ACCUM ALL)

prints the impulse response function related to exogenous (independent) variables. See the "Forecasting" section on page 1756 for details.

The following options can be used in the IMPULSX=( ) option. The options are listed within parentheses.

ACCUM	prints the accumulated impulse response matrices in the transfer function.
ALL	equivalent to specifying both SIMPLE and ACCUM.
SIMPLE	prints the impulse response matrices in the transfer function. This is the default.

PARCOEF
PARCOEF(*number*)

prints the partial autoregression coefficient matrices, Φ_{mm}. With a VAR process, this option is useful for the identification of the order since the Φ_{mm} have the characteristic property that they equal zero for $m > p$ under the hypothetical assumption of a VAR(p) model. See the "Tentative Order Selection" section on page 1761 for details.

PCANCORR
PCANCORR(*number*)

prints the partial canonical correlations of the process at lag m and the test for testing Φ_m=0 for $m > p$. The lag m partial canonical correlations are the canonical correlations between $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, after adjustment for the dependence of these variables on the intervening values $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$. See the "Tentative Order Selection" section on page 1761 for details.

PCORR
PCORR(*number*)

prints the partial correlation matrices. With a VAR process, this option is useful for a tentative order selection by the same property as the partial autoregression coefficient matrices, as described in the PRINT=(PARCOEF) option. See the "Tentative Order Selection" section on page 1761 for details.

ROOTS

prints the eigenvalues of the $kp \times kp$ companion matrix associated with the AR characteristic function $\Phi(B)$, where k is the number of dependent (endogenous) variables, and $\Phi(B)$ is the finite order matrix polynomial in the backshift operator B, such that $B^i\mathbf{y}_t = \mathbf{y}_{t-i}$. These eigenvalues indicate the stationary condition of the process since the stationary condition on the roots of $|\Phi(B)| = 0$ in the VAR(p) model is equivalent to the condition in the corresponding VAR(1) representation that all eigenvalues of the companion matrix be less than one in absolute value. Similarly, you can use this option to check the invertibility of the MA process. In addition, when the GARCH= option is specified, this option prints the roots of the GARCH characteristic polynomials to check covariance stationarity for the GARCH process.

YW

prints Yule-Walker estimates of the preliminary autoregressive model for the dependent (endogenous) variables. The coefficient matrices are printed using the maximum order of the autoregressive process.

Some examples of the PRINT= option are as follows:

```
model y1 y2 / p=1 print=(covy(10) corry(10));
model y1 y2 / p=1 print=(parcoef pcancorr pcorr);
model y1 y2 / p=1 print=(impulse(8) decompose(6) covpe(6));
model y1 y2 / p=1 print=(dynamic roots yw);
```

Lag Specification Options

P= *number*
P= (*number-list***)**

specifies the order of the vector autoregressive process. Subset models of vector autoregressive orders can be specified as, for example, the option P=(1,3,4). the option P=3 is equivalent to the option P=(1,2,3). The default is P=0.

If the option P=0 and there are no exogenous (independent) variables, the AR polynomial order is automatically determined by minimizing an information criterion; if P=0 and the PRIOR= or ECM= option or both is specified the AR polynomial order is determined.

If the ECM= option is specified, subset models of vector autoregressive orders are not allowed and the AR maximum order is used.

```
model y1 y2 / p=3;
model y1 y2 / p=(1,3);
model y1 y2 / p=(1,3) prior;
```

Q= *number*
Q= (*number-list***)**

specifies the order of the moving-average error process. Subset models of moving-average orders can be specified as, for example, the option Q=(1,5). The default is Q=0. The Q= option is ignored when either the GARCH= or XLAG= option is specified.

```
model y1 y2 / p=1 q=1;
model y1 y2 / q=(2);
```

XLAG= *number*
XLAG= (*number-list***)**

specifies the lags of exogenous (independent) variables. Subset models of distributed lags can be specified as, for example, the option XLAG=(2). The default is XLAG=0. To exclude the present values of exogenous (independent) variables from the model, the NOCURRENTX option must be used.

```
model y1 y2 = x1-x3 / xlag=2 nocurrentx;
model y1 y2 = x1-x3 / p=1 xlag=(2);
```

Tentative Order Selection Options

MINIC
MINIC= (TYPE=*value* **P=***number* **Q=***number* **PERROR=***number***)**

prints the information criterion for the appropriate AR and MA tentative order selection and for the diagnostic checks of the fitted model.

If the MINIC= option is not specified, all types of information criteria are printed for diagnostic checks of the fitted model.

The following options can be used in the MINIC=( ) option. The options are listed within parentheses.

P= *number*
P= (p_{min}**:**p_{max}**)**

specifies the range of AR orders. The default is P=(0:5).

PERROR= *number*
PERROR= ($p_{\epsilon,min}$**:**$p_{\epsilon,max}$**)**

specifies the range of AR orders for obtaining the error series. The default is PERROR=($p_{max} : p_{max} + q_{max}$).

Q= *number*
Q= (q_{min}**:**q_{max}**)**

specifies the range of MA orders. The default is Q=(0:5).

TYPE= *value*

specifies the criterion for the model order selection. Valid criteria are as follows:

AIC	specifies the Akaike Information Criterion.
AICC	specifies the Corrected Akaike Information Criterion. This is the default criterion.
FPE	specifies the Final Prediction Error criterion.
HQC	specifies the Hanna-Quinn Criterion.
SBC	specifies the Schwarz Bayesian Criterion. You can also specify this value as TYPE=BIC.

```
model y1 y2 / minic;
model y1 y2 / minic=(type=aic p=5);
```

Cointegration Related Options

There are two options related integrated time series; one is the DFTEST option for a unit root test and the other is the COINTTEST option for a cointegration test.

DFTEST
DFTEST= (DLAG=*number***)**
DFTEST= (DLAG=(*number***) . . . (***number***))**

prints the Dickey-Fuller unit root tests. The option DLAG=(*number*) . . . (*number*) specifies the regular or seasonal unit root test. The *number* should be in {1, 2, 4, 12}. If the *number* is greater than one, a seasonal Dickey-Fuller test is performed. If the

TREND= option is specified, the seasonal unit root test is not available. The default is DLAG=1.

For example, the DFTEST=(DLAG=(1)(12)) option provides two tables, the Dickey-Fuller regular unit root test and the seasonal unit root test.

Some examples of the DFTEST= option are

```
model y1 y2 / p=2 dftest;
model y1 y2 / p=2 dftest=(dlag=4);
model y1 y2 / p=2 dftest=(dlag=(1)(12));
model y1 y2 / p=2 dftest cointtest;
```

COINTTEST
COINTTEST= (JOHANSEN<(=*options*)> SW<(=*options*)> SIGLEVEL=*number*)

The following options can be used in the COINTTEST=( ) option. The options are listed within parentheses.

JOHANSEN
JOHANSEN= (TYPE=*value* IORDER=*number* NORMALIZE=*variable*)

prints the cointegration rank test for multivariate time series based on Johansen method. This test is provided when the number of dependent (endogenous) variables is less than or equal to 11. See the "Vector Error Correction Modeling" section on page 1786 for details.

The VAR(p) model can be written as the error correction model

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

where Π, Φ_i^*, and A are coefficient parameters; D_t is a deterministic term such as a constant, a linear trend, or seasonal dummies.

The $I(1)$ model is defined by one reduced-rank condition. If the cointegration rank is $r < k$, then there exist $k \times r$ matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of rank r such that $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$.

The $I(1)$ model is rewritten as the $I(2)$ model

$$\Delta^2 \mathbf{y}_t = \Pi \mathbf{y}_{t-1} - \Psi \Delta \mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i \Delta^2 \mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

where $\Psi = I_k - \sum_{i=1}^{p-1} \Phi_i^*$ and $\Psi_i = -\sum_{j=i+1}^{p-1} \Phi_i^*$.

The $I(2)$ model is defined by two reduced-rank conditions. One is that $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are $k \times r$ matrices of full-rank r. The other is that $\boldsymbol{\alpha}_\perp' \Psi \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$ where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r) \times s$ matrices with $s \leq k - r$; $\boldsymbol{\alpha}_\perp$ and $\boldsymbol{\beta}_\perp$ are $k \times (k-r)$ matrices of full-rank $k - r$ such that $\boldsymbol{\alpha}'\boldsymbol{\alpha}_\perp = 0$ and $\boldsymbol{\beta}'\boldsymbol{\beta}_\perp = 0$.

The following options can be used in the JOHANSEN=( ) option. The options are listed within parentheses.

IORDER= *number* specifies the integrated order.

IORDER=1	prints the cointegration rank test for an integrated order 1 and prints the long-run parameter, β, and the adjustment coefficient, α. This is the default. If the option IORDER=1 is specified, the AR order should be greater than or equal to 1. When the option P=0, the value of P is temporarily set to 1 for the Johansen test.
IORDER=2	prints the cointegration rank test for integrated orders 1 and 2. If the option IORDER=2 is specified, the AR order should be greater than or equal to 2. If the option P=1, the option IORDER=1 is used; if the option P=0, the value of P is temporarily set to 2.

NORMALIZE= *variable* specifies the dependent (endogenous) variable name whose cointegration vectors are to be normalized. If the normalized variable is different from that specified in the ECM= option or the COINTEG statement, the latter is used.

TYPE= *value* specifies the type of cointegration rank test to be printed. Valid values are as follows:

MAX	prints the cointegration maximum eigenvalue test.
TRACE	prints the cointegration trace test. This is the default.

If the NOINT option is not specified, the procedure prints two different cointegration rank tests in the presence of the unrestricted and restricted deterministic terms (constant or linear trend) models. If the option IORDER=2 is specified, the procedure automatically determines that the option TYPE=TRACE.

```
model y1 y2 / p=2 cointtest=(johansen=(type=max normalize=y1));
model y1 y2 / p=2 cointtest=(johansen=(iorder=2 normalize=y1));
```

SIGLEVEL= *value*

sets the size of cointegration rank tests and common trends tests.
The SIGLEVEL= *value* option must be one of 0.1, 0.05, or 0.01. The default is SIGLEVEL=0.05.

```
model y1 y2 / p=2 cointtest=(johansen siglevel=0.1);
model y1 y2 / p=2 cointtest=(sw siglevel=0.1);
```

SW
SW= (TYPE=*value* **LAG=***number***)**

prints common trends tests for a multivariate time series based on the Stock-Watson method. This test is provided when the number of dependent (endogenous) variables is less than or equal to 6. See the “Common Trends” section on page 1783 for details.

The following options can be used in the SW=( ) option. The options are listed within parentheses.

LAG= *number*	specifies the number of lags. The default is LAG=max(1,p) for the option TYPE=FILTDIF or TYPE=FILTRES, where p is the AR maximum order specified by the P= option; LAG=$O(T^{1/4})$ for the option TYPE=KERNEL, where T is the number of nonmissing observations. If the option LAG= exceeds the default, it is replaced by the default.
TYPE= *value*	specifies the type of common trends test to be printed. Valid values are as follows:

FILTDIF	prints the common trends test based on the filtering method applied to the differenced series. This is the default.
FILTRES	prints the common trends test based on the filtering method applied to the residual series.
KERNEL	prints the common trends test based on the kernel method.

```
model y1 y2 / p=2 cointtest=(sw);
model y1 y2 / p=2 cointtest=(sw=(type=kernel));
```

Bayesian VAR Estimation Options

PRIOR
PRIOR= (MEAN=(*vector***) LAMBDA=***value* **THETA=***value* **IVAR<=(***variables***)>**
NREP=*number* **SEED=***number***)**

specifies the prior value of parameters for the BVAR(p) model. The BVAR model allows the subset model. If the ECM= option is specified with the PRIOR option, the BVECM(p) form is fitted. For the standard errors of the predictors, the bootstrap procedure is used. See the “Bayesian VAR Modeling” section on page 1773 for details.

The following options can be used in the PRIOR=( ) option. The options are listed within parentheses.

IVAR
IVAR= (*variables***)**

specifies an integrated BVAR(p) model. If you use the IVAR option without *variables*, it sets the overall prior mean of the first lag of each variable equal to one in its own equation and sets all other coefficients to zero. If *variables* are specified, it sets the prior mean of the first lag of the specified variables equal to one in its own equation and sets all other coefficients to zero. When the series $\mathbf{y}_t = (y_1, y_2)'$ follows a bivariate BVAR(2) process, the IVAR or IVAR=(y_1 y_2) option is equivalent to specifying MEAN=(1 0 0 0 0 1 0 0).

If the PRIOR=(MEAN=) or ECM= option is specified, the IVAR= option is ignored.

LAMBDA= *value*

specifies the prior standard deviation of the AR coefficient parameter matrices. It should be a positive number. The default is LAMBDA=1. As the value of the option LAMBDA= is larger, a BVAR(p) model is close to a VAR(p) model.

MEAN= (*vector***)**

specifies the mean vector in the prior distribution for the AR coefficients. If the vector is not specified, the prior value is assumed to be a zero vector. See the "Bayesian VAR Modeling" section on page 1773 for details.

You can specify the mean vector by order of the equation. Let $(\delta, \Phi_1, ..., \Phi_p)$ be the parameter sets to be estimated and $\Phi = (\Phi_1, ..., \Phi_p)$ be the AR parameter sets. Then the mean vector is specified by row-wise from the Φ; that is, the option MEAN=($\text{vec}(\Phi')$).

For the PRIOR=(mean) option in the BVAR(2),

$$\Phi = \left(\begin{array}{cccc} \phi_{1,11} & \phi_{1,12} & \phi_{2,11} & \phi_{2,12} \\ \phi_{1,21} & \phi_{1,22} & \phi_{2,21} & \phi_{2,22} \end{array} \right) = \left(\begin{array}{cccc} 2 & 0.1 & 1 & 0 \\ 0.5 & 3 & 0 & -1 \end{array} \right)$$

$$\mathbf{model\ \ y1\ \ y2\ \ /\ \ p = 2\ \ prior} = (\mathbf{mean} = (\ \mathbf{2\ \ 0.1\ \ 1\ \ 0\ \ 0.5\ \ 3\ \ 0\ \ -1}));$$

The deterministic terms are considered to shrink toward zero; you must omit prior means of deterministic terms such as a constant, seasonal dummies, or trends.

For a Bayesian error correction model, you specify a mean vector for only lagged AR coefficients, Φ_i^*, in terms of regressors $\Delta\mathbf{y}_{t-i}$, for $i = 1, ..., (p-1)$ in the VECM(p) representation. The diffused prior variance of $\boldsymbol{\alpha}$ is used since $\boldsymbol{\beta}$ is replaced by $\hat{\boldsymbol{\beta}}$ estimated in a nonconstrained VECM(p) form.

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\mathbf{z}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta\mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

where $\mathbf{z}_t = \boldsymbol{\beta}'\mathbf{y}_t$.

For example, in the case of a bivariate ($k = 2$) BVECM(2) form, the option

$$\mathrm{MEAN} = (\phi^*_{1,11}\ \phi^*_{1,12}\ \phi^*_{1,21}\ \phi^*_{1,22})$$

where $\phi^*_{1,ij}$ is the (i,j)th element of the matrix Φ^*_1.

NREP= *number*
specifies the number of bootstrap replications. The default is NREP=100.

SEED= *number*
specifies seeds to generate uniform random numbers for resampling. By default, the system clock is used to generate the random seed.

THETA= *value*
specifies the prior standard deviation of the AR coefficient parameter matrices. The *value* is in the interval (0,1). The default is THETA=0.1. As the value of the THETA= option is close to 1, a BVAR(p) model is close to a VAR(p) model.

Some examples of the PRIOR= option are

```
model y1 y2 / p=2 prior;
model y1 y2 / p=2 prior=(theta=0.2 lambda=5);
```

Vector Error Correction Model Options

ECM=(RANK=*number* **NORMALIZE=***variable* **ECTREND)**
specifies a vector error correction model.

The following options can be used in the ECM=( ) option. The options are listed within parentheses.

NORMALIZE= *variable*
specifies a single dependent variable name whose cointegrating vectors are normalized. If the variable name is different from that specified in the COINTEG statement, the latter is used.

RANK= *number*
specifies the cointegration rank. This option is required in the ECM= option. The value of the RANK= option should be greater than zero and less than or equal to the number of dependent (endogenous) variables, k. If the rank is different from that specified in the COINTEG statement, the latter is used.

ECTREND
specifies the restriction on the drift in the VECM(p) form.

- There is no separate drift in the VECM(p) form, but a constant enters only through the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \beta_0)(\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi^*_i \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

$$\mathbf{model\ \ y1\ \ y2\ \ /\ \ p = 2\ \ ecm = (rank = 1\ \ ectrend);}$$

- There is a separate drift and no separate linear trend in the VECM(p) form, but a linear trend enters only through the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \beta_1)(\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

$$\mathbf{model\ \ y1\ \ y2\ \ /\ \ p = 2\ \ ecm = (rank = 1\ \ ectrend)\ \ trend = linear};$$

If the NSEASON option is specified, then the NSEASON option is ignored; if the NOINT option is specified, then the ECTREND option is ignored.

Some examples of the ECM= option are

```
model y1 y2 / p=2 ecm=(rank=1 normalized=y1);
model y1 y2 / p=2 ecm=(rank=1 ectrend) trend=linear;
```

GARCH Model Estimation Options

GARCH=(Q=*number* **P=***number* **FORM=** *value* **MEAN)** Experimental

specifies a GARCH-type multivariate conditional heteroscedasticity model. The GARCH= option in the MODEL statement specifies the family of GARCH models to be estimated. See the "Multivariate GARCH Modeling" section on page 1804 for details.

The following options can be used in the GARCH= option.

FORM= *value*

specifies the representation for a GARCH model. Valid values are as follows:

BEKK	specifies a *BEKK* representation. This is the default.
BEW	specifies a vectorized representation.
DIAG	specifies a diagonal representation.

MEAN

specifies the GARCH-M model.

P=*number*
P=(*number-list***)**

specifies the order of the process or the subset of GARCH terms to be fitted. By default, P=0.

Q=*number*
Q=(*number-list***)**

specifies the order of the process or the subset of ARCH terms to be fitted. This option is required in the GARCH= option.

For the VAR(1)-ARCH(1) model,

```
model y1 y2 / p=1 garch=(q=1);
```

For the multivariate GARCH(1,1) model,

```
   model y1 y2 / garch=(q=1 p=1 form=diag);
```

For the multivariate ARCH(1)-M model,

```
   model y1 y2 / garch=(q=1 mean);
```

Other multivariate GARCH-type models are

```
   model y1 y2 = x1 / garch=(q=1 mean);
   model y1 y2 = x1 / p=1 xlag=1 garch=(q=1 form=bew);
```

NLOPTIONS Statement

NLOPTIONS *options* **;**

PROC VARMAX uses the NonLinear Optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options of the NLOPTIONS statement, see Chapter 10, "Nonlinear Optimization Methods."

An example of the NLOPTIONS statement is

```
   proc varmax data=one;
      nloptions tech=qn;
      model y1 y2 / p=2;
   run;
```

OUTPUT Statement

OUTPUT < *options* >**;**

The OUTPUT statement generates and prints forecasts based on the model estimated in the previous MODEL statement and, optionally, creates an output SAS data set that contains these forecasts.

ALPHA= *value*

sets the forecast confidence limits. The option ALPHA=*value* must be between 0 and 1. When you specify the option ALPHA=α, the upper and lower confidence limits define the $1 - \alpha$ confidence interval. The default is ALPHA=0.05, which produces 95% confidence intervals.

BACK= *number*

specifies the number of observations before the end of the data at which the multistep-ahead forecasts are to begin. The BACK= option value must be less than or equal to the number of observations minus the number of lagged regressors in the model. The default is BACK=0, which means that the forecast starts at the end of the available data.

LEAD= *number*
specifies the number of multistep-ahead forecast values to compute. The default is LEAD=12.

NOPRINT
suppresses the printed forecast values of each dependent (endogenous) variable.

OUT= *SAS-data-set*
writes the forecast values to an output data set.

Some examples of the OUTPUT statements are

```
proc varmax data=one;
   model y1 y2 / p=2;
   output lead=6 back=2;
run;

proc varmax data=one;
   model y1 y2 / p=2;
   output out=for noprint;
run;
```

RESTRICT Statement

RESTRICT *restriction ... restriction* **;**

The RESTRICT statement restricts the specified parameters to the specified values. Only one RESTRICT statement is allowed.

The *restriction*'s form is *parameter = value* and each restriction is separated by commas. Parameters are referred by the following keywords:

- CONST(i) is the intercept parameter of the current value ith time series y_{it}
- AR(l,i,j) is the autoregressive parameter of the previous lag l value of the jth dependent (endogenous) variable, $y_{j,t-l}$, to the ith dependent variable at time t, y_{it}
- MA(l,i,j) is the moving-average parameter of the previous lag l value of the jth error process, $\boldsymbol{\epsilon}_{j,t-l}$, to the ith dependent variable at time t, y_{it}
- XL(l,i,j) is the exogenous parameter of the previous lag l value of the jth exogenous (independent) variable, $x_{j,t-l}$, to the ith dependent variable at time t, y_{it}
- SDUMMY(i,j) is the jth seasonal dummy of the ith time series at time t, y_{it}, where $j = 1, \ldots, (nseason - 1)$
- LTREND(i) is the linear trend parameter of the current value ith time series y_{it}
- QTREND(i) is the quadratic trend parameter of the current value ith time series y_{it}

The following keywords are for the fitted GARCH model. The indexes i and j refer to the position of the element in the coefficient matrix.

- GCHM(i,j) is the GARCH-M parameter of $\mathrm{vech}(H_t)$ for $i = 1, \ldots, k$ and $j = 1, \ldots, k(k+1)/2$
- GCHC(i,j) is the constant parameter of the covariance matrix, H_t, for $1 \le i \le j \le k$
- ACH(l,i,j) is the ARCH parameter of the previous lag l value of $\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t'$
- GCH(l,i,j) is the GARCH parameter of the previous lag l value of covariance matrix, H_t

The indexes i and j for ACH(l,i,j) and GCH(l,i,j) have different ranges according to the GARCH model representation. For example, $i, j = 1, \ldots, k$ for a *BEKK* representation, $i, j = 1, \ldots, k(k+1)/2$ for a *BEW* representation, and $i = j = 1, \ldots, k(k+1)/2$ for a *diagonal* representation.

To use the RESTRICT statement, you need to know the form of the model. If you do not specify any order of the model, the RESTRICT statement is not applicable.

Restricted parameter estimates are computed by introducing a Lagrangian parameter for each restriction (Pringle and Raynor 1971). The Lagrangian parameter measures the sensitivity of the sum of square errors to the restriction. The estimates of these Lagrangian parameters and their significance are printed in the restriction results table.

The following are examples of the RESTRICT statement. The first example shows a bivariate (k=2) VAR(2) model,

```
proc varmax data=one;
   model y1 y2 / p=2;
   restrict AR(1,1,2)=0, AR(2,1,2)=0.3;
run;
```

The following shows a bivariate (k=2) VARX(1,1) model with three exogenous variables,

```
proc varmax data=two;
   model y1 = x1 x2, y2 = x2 x3 / p=1 xlag=1;
   restrict XL(0,1,1)=-1.2, XL(1,2,3)=0;
run;
```

TEST Statement

TEST *restriction ... restriction* **;**

The TEST statement performs the Wald test for the joint hypothesis specified in the statement. The *restriction*'s form is *parameter* = *value* and each restriction is separated by commas. The *restriction*'s form is referred to by the same rule in the RESTRICT statement. Any number of TEST statements can be specified.

To use the TEST statement, you need to know the form of the model. If you do not specify any order of the model, the TEST statement is not applicable.

See the “Granger-Causality Test” section on page 1769 for the Wald test.

The following is an example of the TEST statement. In case of a bivariate (k=2) VAR(2) model,

```
proc varmax data=one;
   model y1 y2 / p=2;
   test AR(1,1,2)=0, AR(2,1,2)=0;
run;
```

Details

Missing Values

The VARMAX procedure currently does not support missing values. The procedure uses the first contiguous group of observations with no missing values for any of the MODEL statement variables. Observations at the beginning of the data set with missing values for any MODEL statement variables are not used or included in the output data set. At the end of the data set, observations can have dependent (endogenous) variables with missing values and independent (exogenous)variables with nonmissing values.

VARMAX Modeling

The vector autoregressive moving-average model with exogenous variables is called the VARMAX(p,q,s) model. The form of the model can be written as

$$\mathbf{y}_t = \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

where the output variables of interest, $\mathbf{y}_t = (y_{1t}, \ldots, y_{kt})'$, can be influenced by other input variables, $\mathbf{x}_t = (x_{1t}, \ldots, x_{rt})'$, which are determined outside the system of interest. The variables $\mathbf{y}_t$ are referred to as dependent, response, or endogenous variables, and the variables $\mathbf{x}_t$ are referred to as independent, input, predictor, regressor, or exogenous variables. The unobserved noise variables, $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \ldots, \epsilon_{kt})'$, are a vector white noise process.

The VARMAX(p,q,s) model can be written

$$\Phi(B)\mathbf{y}_t \quad = \quad \Theta^*(B)\mathbf{x}_t + \Theta(B)\epsilon_t$$

where

$$\begin{aligned} \Phi(B) &= I_k - \Phi_1 B - \cdots - \Phi_p B^p \\ \Theta^*(B) &= \Theta_0^* + \Theta_1^* B + \cdots + \Theta_s^* B^s \\ \Theta(B) &= I_k - \Theta_1 B - \cdots - \Theta_q B^q \end{aligned}$$

are matrix polynomials in B in the backshift operator, such that $B^i \mathbf{y}_t = \mathbf{y}_{t-i}$, the Φ_i and Θ_i are $k \times k$ matrices, and the Θ_i^* are $k \times r$ matrices.

The following assumptions are made:

- $\mathrm{E}(\boldsymbol{\epsilon}_t) = 0$, $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$, which is positive-definite, and $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s') = 0$ for $t \neq s$.
- For stationarity and invertibility of the VARMAX process, the roots of $|\Phi(z)| = 0$ and $|\Theta(z)| = 0$ are outside the unit circle.
- The exogenous (independent) variables $\mathbf{x}_t$ are not correlated with residuals $\boldsymbol{\epsilon}_t$, $\mathrm{E}(\mathbf{x}_t \boldsymbol{\epsilon}_t') = 0$. The exogenous variables can be stochastic or nonstochastic. When the exogenous variables are stochastic and their future values are unknown, then forecasts of these future values are needed in the forecasting of the future values of the endogenous variables. On occasion, future values of the exogenous variables can be assumed to be known because they are deterministic variables. Note that the VARMAX procedure assumes that the exogenous variables are nonstochastic if future values are available in the input data set. Otherwise, the exogenous variables are assumed to be stochastic and their future values are forecasted by assuming that they follow the VARMA(p,q) model.

State-Space Modeling

Another representation of the VARMAX(p,q,s) model is in the form of a state-variable or a state-space model, which consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + K\mathbf{x}_t + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \\ \mathbf{x}_t \\ \vdots \\ \mathbf{x}_{t-s+1} \\ \boldsymbol{\epsilon}_t \\ \vdots \\ \boldsymbol{\epsilon}_{t-q+1} \end{bmatrix}, \quad K = \begin{bmatrix} \Theta_0^* \\ 0_{k\times r} \\ \vdots \\ 0_{k\times r} \\ I_r \\ 0_{r\times r} \\ \vdots \\ 0_{r\times r} \\ 0_{k\times r} \\ \vdots \\ 0_{k\times r} \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ 0_{k\times k} \\ \vdots \\ 0_{k\times k} \\ 0_{r\times k} \\ \vdots \\ 0_{r\times k} \\ I_{k\times k} \\ 0_{k\times k} \\ \vdots \\ 0_{k\times k} \end{bmatrix}$$

$$F = \begin{bmatrix} \Phi_1 & \cdots & \Phi_{p-1} & \Phi_p & \Theta_1^* & \cdots & \Theta_{s-1}^* & \Theta_s^* & -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ I_k & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & I_k & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & I_r & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & I_r & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

and

$$H = [I_k, 0_{k\times k}, \ldots, 0_{k\times k}, 0_{k\times r}, \ldots, 0_{k\times r}, 0_{k\times k}, \ldots, 0_{k\times k}]$$

On the other hand, it is assumed that $\mathbf{x}_t$ follows a VARMA(p,q) model

$$\mathbf{x}_t = \sum_{i=1}^{p} A_i \mathbf{x}_{t-i} + \mathbf{a}_t - \sum_{i=1}^{q} C_i \mathbf{a}_{t-i}$$

or $A(B)\mathbf{x}_t = C(B)\mathbf{a}_t$, where $A(B) = I_r - A_1 B - \cdots - A_p B^p$ and $C(B) = I_r - C_1 B - \cdots - C_q B^q$ are matrix polynomials in B, and the A_i and C_i are $r \times r$ matrices. Without loss of generality, the AR and MA orders can be taken to be the same as the VARMAX(p,q,s) model, and $\mathbf{a}_t$ and $\boldsymbol{\epsilon}_t$ are independent white noise processes.

Under suitable (such as stationarity) conditions, $\mathbf{x}_t$ is represented by an infinite order moving-average process

$$\mathbf{x}_t = A(B)^{-1}C(B)\mathbf{a}_t = \Psi^x(B)\mathbf{a}_t = \sum_{j=0}^{\infty} \Psi_j^x \mathbf{a}_{t-j}$$

where $\Psi^x(B) = A(B)^{-1}C(B) = \sum_{j=0}^{\infty} \Psi_j^x B^j$.

The optimal (Minimum Mean Squared Error, MMSE) i-step-ahead forecast of $\mathbf{x}_{t+i}$ is

$$\begin{aligned} \mathbf{x}_{t+i|t} &= \sum_{j=i}^{\infty} \Psi_j^x \mathbf{a}_{t+i-j} \\ \mathbf{x}_{t+i|t+1} &= \mathbf{x}_{t+i|t} + \Psi_{i-1}^x \mathbf{a}_{t+1} \end{aligned}$$

For $i > q$,

$$\mathbf{x}_{t+i|t} = \sum_{j=1}^{p} A_j \mathbf{x}_{t+i-j|t}$$

The VARMAX(p,q,s) model has an absolutely convergent representation as

$$\begin{aligned}
\mathbf{y}_t &= \Phi(B)^{-1}\Theta^*(B)\mathbf{x}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\
&= \Psi^*(B)\Psi^x(B)\mathbf{a}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\
&= V(B)\mathbf{a}_t + \Psi(B)\boldsymbol{\epsilon}_t
\end{aligned}$$

or

$$\mathbf{y}_t = \sum_{j=0}^{\infty} V_j \mathbf{a}_{t-j} + \sum_{j=0}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t-j}$$

where $\Psi(B) = \Phi(B)^{-1}\Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$, $\Psi^*(B) = \Phi(B)^{-1}\Theta^*(B)$, and $V(B) = \Psi^*(B)\Psi^x(B) = \sum_{j=0}^{\infty} V_j B^j$.

The optimal (MMSE) i-step-ahead forecast of $\mathbf{y}_{t+i}$ is

$$\begin{aligned}
\mathbf{y}_{t+i|t} &= \sum_{j=i}^{\infty} V_j \mathbf{a}_{t+i-j} + \sum_{j=i}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+i-j} \\
\mathbf{y}_{t+i|t+1} &= \mathbf{y}_{t+i|t} + V_{i-1}\mathbf{a}_{t+1} + \Psi_{i-1}\boldsymbol{\epsilon}_{t+1}
\end{aligned}$$

for $i = 1, \ldots, v$ with $v = \max(p, q+1)$. For $i > q$,

$$\begin{aligned}
\mathbf{y}_{t+i|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \mathbf{x}_{t+i|t} + \sum_{j=1}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \sum_{j=1}^{p} A_j \mathbf{x}_{t+i-j|t} + \sum_{j=1}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=1}^{u} (\Theta_0^* A_j + \Theta_j^*) \mathbf{x}_{t+i-j|t}
\end{aligned}$$

where $u = \max(p, s)$.

Define $\Pi_j = \Theta_0^* A_j + \Theta_j^*$. For $i = v > q$ with $v = \max(p, q+1)$, you obtain

$$\begin{aligned}
\mathbf{y}_{t+v|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^{u} \Pi_j \mathbf{x}_{t+v-j|t} \ \text{ for } \ u \le v \\
\mathbf{y}_{t+v|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^{r} \Pi_j \mathbf{x}_{t+v-j|t} \ \text{ for } \ u > v
\end{aligned}$$

From the preceding relations, a state equation is

$$\mathbf{z}_{t+1} = F\mathbf{z}_t + K\mathbf{x}_t^* + G\mathbf{e}_{t+1}$$

and an observation equation is

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \mathbf{y}_{t+1|t} \\ \vdots \\ \mathbf{y}_{t+v-1|t} \\ \mathbf{x}_t \\ \mathbf{x}_{t+1|t} \\ \vdots \\ \mathbf{x}_{t+v-1|t} \end{bmatrix}, \quad \mathbf{x}_t^* = \begin{bmatrix} \mathbf{x}_{t+v-u} \\ \mathbf{x}_{t+v-u+1} \\ \vdots \\ \mathbf{x}_{t-1} \end{bmatrix}, \quad \mathbf{e}_{t+1} = \begin{bmatrix} \mathbf{a}_{t+1} \\ \boldsymbol{\epsilon}_{t+1} \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & I_k & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I_k & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi_v & \Phi_{v-1} & \Phi_{v-2} & \cdots & \Phi_1 & \Pi_v & \Pi_{v-1} & \Pi_{v-2} & \cdots & \Pi_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & I_r & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & I_r & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_v & A_{v-1} & A_{v-2} & \cdots & A_1 \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Pi_u & \Pi_{u-1} & \cdots & \Pi_{v+1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad G = \begin{bmatrix} V_0 & I_k \\ V_1 & \Psi_1 \\ \vdots & \vdots \\ V_{v-1} & \Psi_{v-1} \\ I_r & 0_{r\times k} \\ \Psi_1^x & 0_{r\times k} \\ \vdots & \vdots \\ \Psi_{v-1}^x & 0_{r\times k} \end{bmatrix}$$

and

$$H = [I_k, 0_{k\times k}, \ldots, 0_{k\times k}, 0_{k\times r}, \ldots, 0_{k\times r}]$$

Note that the matrix K and the input vector $\mathbf{x}_t^*$ are defined only when $u > v$.

Dynamic Simultaneous Equations Modeling

In the econometrics literature, the VARMAX(p,q,s) model could be written in the following slightly different form, which is referred to as a *dynamic simultaneous equations* model or a *dynamic structural equations* model.

Since $\mathrm{E}(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t') = \Sigma$ is assumed to be positive-definite, there exists a lower triangular matrix A_0 with ones on the diagonals such that $A_0\Sigma A_0' = \Sigma^d$, where Σ^d is a diagonal matrix with positive diagonal elements.

$$A_0\mathbf{y}_t = \sum_{i=1}^{p} A_i\mathbf{y}_{t-i} + \sum_{i=0}^{s} C_i^*\mathbf{x}_{t-i} + C_0\boldsymbol{\epsilon}_t - \sum_{i=1}^{q} C_i\boldsymbol{\epsilon}_{t-i}$$

where $A_i = A_0\Phi_i$, $C_i^* = A_0\Theta_i^*$, $C_0 = A_0$, and $C_i = A_0\Theta_i$.

As an alternative form,

$$A_0\mathbf{y}_t = \sum_{i=1}^{p} A_i\mathbf{y}_{t-i} + \sum_{i=0}^{s} C_i^*\mathbf{x}_{t-i} + \mathbf{a_t} - \sum_{\mathbf{i=1}}^{\mathbf{q}} \mathbf{C_i a_{t-i}}$$

where $A_i = A_0\Phi_i$, $C_i^* = A_0\Theta_i^*$, $C_i = A_0\Theta_i A_0^{-1}$, and $\mathbf{a_t} = \mathbf{C_0}\boldsymbol{\epsilon}_\mathbf{t}$ has a diagonal covariance matrix Σ^d. The PRINT=(DYNAMIC) option follows this form.

A dynamic simultaneous equations model involves a leading (lower triangular) coefficient matrix for $\mathbf{y}_t$ at lag 0 or a leading coefficient matrix for $\boldsymbol{\epsilon}_t$ at lag 0. Such a representation of the VARMAX(p,q,s) model can be more useful in certain circumstances than the standard representation. From the linear combination of the dependent variables obtained by $A_0\mathbf{y}_t$, you can easily see the relationship between the dependent variables in the current time.

The following statements provide the dynamic simultaneous equations of the VAR(1) model.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(dynamic);
run;
```

```
                          The VARMAX Procedure

                       Covariances of Innovations
                          of the Dynamic Model

                 Variable              y1              y2

                 y1               1.28875         0.00000
                 y2               0.00000         1.29578


                     Dynamic AR Coefficient Estimates

               Lag    Variable              y1              y2

                 0    y1               1.00000         0.00000
                      y2              -0.30845         1.00000
                 1    y1               1.15977        -0.51058
                      y2               0.18861         0.54247


                    Dynamic Model Parameter Estimates

                                            Standard
Equation Parameter       Estimate              Error t Value Pr > |t| Variable

y1       AR1_1_1          1.15977       0.05508   21.06   0.0001 y1(t-1)
         AR1_1_2         -0.51058       0.07140   -7.15   0.0001 y2(t-1)
y2       AR0_2_1          0.30845                                y1(t)
         AR1_2_1          0.18861       0.05779    3.26   0.0015 y1(t-1)
         AR1_2_2          0.54247       0.07491    7.24   0.0001 y2(t-1)
```

Figure 30.25. Dynamic Simultaneous Equations (DYNAMIC Option)

In Figure 30.4 on page 1699, the covariance of ϵ_t is

$$\Sigma_{\boldsymbol{\epsilon}} = \left(\begin{array}{rr} 1.28875 & 0.39751 \\ 0.39751 & 1.41839 \end{array} \right)$$

By the decomposition of $\Sigma_{\boldsymbol{\epsilon}}$, you get a diagonal matrix ($\Sigma_{\mathbf{a}}$) and a lower triangular matrix (A_0) such as $\Sigma_{\mathbf{a}} = A_0 \Sigma_{\boldsymbol{\epsilon}} A_0'$ where

$$\Sigma_{\mathbf{a}} = \left(\begin{array}{rr} 1.28875 & 0 \\ 0 & 1.29578 \end{array} \right) \text{ and } A_0 = \left(\begin{array}{rr} 1 & 0 \\ -0.30845 & 1 \end{array} \right)$$

The simultaneous equations model is written as

$$\left(\begin{array}{rr} 1 & 0 \\ -0.30845 & 1 \end{array} \right) \mathbf{y}_t = \left(\begin{array}{rr} 1.15977 & -0.51058 \\ 0.18861 & 0.54247 \end{array} \right) \mathbf{y}_{t-1} + \mathbf{a_t}$$

Two univariate equations can be written as

$$\begin{aligned} y_{1t} &= 1.15977 y_{1,t-1} - 0.51058 y_{2,t-1} + a_{1t} \\ y_{2t} &= 0.30845 y_{1t} + 0.18861 y_{1,t-1} + 0.54247 y_{2,t-1} + a_{2t} \end{aligned}$$

Impulse Response Function

The VARMAX(p,q,s) model has a convergent representation

$$\mathbf{y}_t = \Psi^*(B)\mathbf{x}_t + \Psi(B)\boldsymbol{\epsilon}_t$$

where $\Psi^*(B) = \Phi(B)^{-1}\Theta^*(B) = \sum_{j=0}^{\infty} \Psi_j^* B^j$ and $\Psi(B) = \Phi(B)^{-1}\Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$.

The elements of the matrices Ψ_j from the operator $\Psi(B)$, called the impulse response, can be interpreted as the impact that a shock in one variable has on another variable. Let $\psi_{j,in}$ be the *element* of the Ψ_j. The notation i is the index for the impulse variable, and n is the index for the response variable (impulse $\rightarrow$ response). For instance, $\psi_{j,11}$ is an impulse response to $y_{1t} \rightarrow y_{1t}$, and $\psi_{j,12}$ is an impulse response to $y_{1t} \rightarrow y_{2t}$.

The accumulated impulse response function is the cumulative sum of the impulse response function, $\Psi_l^a = \sum_{j=0}^{l} \Psi_j$.

The MA representation with a standardized white noise innovation process offers a further possibility to interpret a VARMA(p,q) model. Since Σ is positive-definite, there is a lower triangular matrix P such that $\Sigma = PP'$. The alternate MA representation is written as

$$\mathbf{y}_t = \Psi^o(B)\mathbf{u}_t$$

where $\Psi^o(B) = \sum_{j=0}^{\infty} \Psi_j^o B^j$, $\Psi_j^o = \Psi_j P$, and $\mathbf{u}_t = P^{-1}\boldsymbol{\epsilon}_t$.

The elements of the matrices Ψ_j^o, called the *orthogonal impulse response*, can be interpreted as the effects of the components of the standardized shock process $\mathbf{u}_t$ on the process $\mathbf{y}_t$ at the lag j.

The coefficient matrix Ψ_j^* from the transfer function operator $\Psi^*(B)$ can be interpreted as the effects that changes in the exogenous variables $\mathbf{x}_t$ have on the output variable $\mathbf{y}_t$ at the lag j, and is called an impulse response matrix in the transfer function.

The accumulated impulse response in the transfer function is the cumulative sum of the impulse response in the transfer function, $\Psi_l^{*a} = \sum_{j=0}^{l} \Psi_j^*$.

The asymptotic distributions of the impulse functions can be seen in the "VAR Modeling" section on page 1766.

The following statements provide the impulse response and the accumulated impulse response in the transfer function for a VARX(1,0) model. Parts of the VARMAX procedure output are shown in Figure 30.26 and Figure 30.27

```
proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1 print=(impulsx=(all)) lagmax=15
                          printform=univariate;
run;
```

```
                           The VARMAX Procedure

                  Simple Impulse Response of Transfer
                         Function by Variable

        Variable        Lag            x1              x2

        y1                0       1.69281        -0.00859
                          1       0.35399         0.01727
                          2       0.09090         0.00714
                          :          :               :
                         14       0.03319         0.00024
                         15       0.03195         0.00023
        y2                0      -6.09850         2.57980
                          1      -5.15484         0.45445
                          2      -3.04168         0.04391
                          :          :               :
                         14      -1.32641        -0.00966
                         15      -1.27682        -0.00930
        y3                0      -0.02317        -0.01274
                          1       1.57476        -0.01435
                          2       1.80231         0.00398
                          :          :               :
                         14       1.16268         0.00846
                         15       1.11921         0.00815
```

Figure 30.26. Impulse Response in Transfer Function (IMPULSX= Option)

In Figure 30.26, the variables $x1$ and $x2$ are impulses and the variables $y1$, $y2$, and $y3$ are responses. You can read the table matching the pairs of $impulse \rightarrow response$ such as $x1 \rightarrow y1$, $x1 \rightarrow y2$, $x1 \rightarrow y3$, $x2 \rightarrow y1$, $x2 \rightarrow y2$, and $x2 \rightarrow y3$. In the pair of $x1 \rightarrow y1$, you can see the long-run responses of $y1$ to an impulse in $x1$ (the values are 1.69281, 0.35399, 0.09090,... for lag 0, lag 1, lag 2,...).

```
                              The VARMAX Procedure

                        Accumulated Impulse Response of
                         Transfer Function by Variable

              Variable         Lag              x1              x2

              y1                 0         1.69281        -0.00859
                                 1         2.04680         0.00868
                                 2         2.13770         0.01582
                                 :            :               :
                                14         2.63183         0.02162
                                15         2.66378         0.02185
              y2                 0        -6.09850         2.57980
                                 1       -11.25334         3.03425
                                 2       -14.29502         3.07816
                                 :            :               :
                                14       -34.35946         2.93139
                                15       -35.63628         2.92210
              y3                 0        -0.02317        -0.01274
                                 1         1.55159        -0.02709
                                 2         3.35390        -0.02311
                                 :            :               :
                                14        20.71402         0.10051
                                15        21.83323         0.10866
```

Figure 30.27. Accumulated Impulse Response in Transfer Function (IMPULSX= Option)

The following statements provide the impulse response function, the accumulated impulse response function, and the orthogonalized impulse response function with their standard errors for a VAR(1) model. Parts of the VARMAX procedure output are shown in Figure 30.28, Figure 30.29, and Figure 30.30.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=15 print=(impulse=(all))
                 printform=univariate;
run;
```

```
                              The VARMAX Procedure

                      Simple Impulse Response by Variable

               Variable     Lag                y1              y2

               y1           1             1.15977        -0.51058
                            STD           0.05508         0.05898
                            2             1.06612        -0.78872
                            STD           0.10450         0.10702
                            :                :               :
                            14            0.08202         0.02870
                            STD           0.10579         0.09483
                            15            0.11080        -0.03083
                            STD           0.09277         0.08778
               y2           1             0.54634         0.38499
                            STD           0.05779         0.06188
                            2             0.84396        -0.13073
                            STD           0.08481         0.08556
                            :                :               :
                            14           -0.03071         0.12557
                            STD           0.10081         0.09391
                            15            0.03299         0.06403
                            STD           0.09375         0.08487
```

Figure 30.28. Impulse Response Function (IMPULSE= Option)

Figure 30.28 is the part of the output in a univariate format associated with the PRINT=(IMPULSE=) option for the impulse response function. The keyword STD stands for the standard errors of the elements. The matrix in terms of the lag 0 does not print since it is the identity. In Figure 30.28, the horizontal variables $y1$ and $y2$ are impulses and the vertical variables $y1$ and $y2$ are responses. You can read the table matching the pairs of $impulse(horizontal) \rightarrow response(vertical)$ such as $y1 \rightarrow y1$, $y1 \rightarrow y2$, $y2 \rightarrow y1$, and $y2 \rightarrow y2$. For example, in the pair of $y1 \rightarrow y1$, you can see the long-run responses of $y1$ to an impulse in itself. The first two lags are significant (the value of lag 1 is 1.15977 with the standard error 0.05508 and the value of lag 2 is 1.06612 with the standard error 0.10450), but the last two lags are insignificant (the value of lag 14 is 0.08202 with the standard error 0.10579 and the value of lag 15 is 0.11080 with the standard error 0.09277).

```
                              The VARMAX Procedure

                   Accumulated Impulse Response by Variable

               Variable     Lag                y1              y2

               y1           1             2.15977        -0.51058
                            STD           0.05508         0.05898
                            2             3.22589        -1.29929
                            STD           0.21684         0.22776
                            :               :               :
                            14            3.11982        -2.53992
                            STD           1.90364         1.84193
                            15            3.23062        -2.57074
                            STD           1.57743         1.52719
               y2           1             0.54634         1.38499
                            STD           0.05779         0.06188
                            2             1.39030         1.25426
                            STD           0.17614         0.18392
                            :               :               :
                            14            2.71782        -0.73442
                            STD           2.57030         2.32369
                            15            2.75080        -0.67040
                            STD           2.08022         1.96462
```

Figure 30.29. Accumulated Impulse Response Function (IMPULSE= Option)

Figure 30.29 is the part of the output in a univariate format associated with the PRINT=(IMPULSE=) option for the accumulated impulse response function. The matrix in terms of the lag 0 does not print since it is the identity.

```
                          The VARMAX Procedure

            Orthogonalized Impulse Response by Variable

               Variable     Lag                y1              y2

               y1           0             1.13523         0.00000
                            STD           0.08068         0.00000
                            1             1.13783        -0.58120
                            STD           0.10666         0.14110
                            2             0.93412        -0.89782
                            STD           0.13113         0.16776
                            :               :               :
                            3             0.61756        -0.96528
                            14            0.10316         0.03267
                            STD           0.09791         0.10849
                            15            0.11499        -0.03509
                            STD           0.08426         0.10065
               y2           0             0.35016         1.13832
                            STD           0.11676         0.08855
                            1             0.75503         0.43824
                            STD           0.06949         0.10937
                            2             0.91231        -0.14881
                            STD           0.10553         0.13565
                            :               :               :
                            14            0.00910         0.14294
                            STD           0.09504         0.10739
                            15            0.05987         0.07288
                            STD           0.08779         0.09695
```

Figure 30.30. Orthogonalized Impulse Response Function (IMPULSE= Option)

Figure 30.30 is the part of the output in a univariate format associated with the PRINT=(IMPULSE=) option for the orthogonalized impulse response function. The horizontal variables $y1\_innovation$ and $y2\_innovation$ are impulses and the vertical variables $y1$ and $y2$ are responses. You can read the table matching the pairs of $impulse(horizontal) \rightarrow response(vertical)$ such as $y1\_innovation \rightarrow y1$, $y1\_innovation \rightarrow y2$, $y2\_innovation \rightarrow y1$, and $y2\_innovation \rightarrow y2$.

In Figure 30.4 on page 1699, there is a positive correlation between ε_{1t} and ε_{2t}. A shock in $y1$ may be accompanied by a shock in $y2$ in the same period. For example, in the pair of $y1\_innovation \rightarrow y2$, you can see the long-run responses of $y2$ to an impulse in $y1\_innovation$. The first three lags are significant (the value of lag 0 is 0.35016 with the standard error 0.11676, the value of lag 1 is 0.75503 with the standard error 0.06949, and the value of lag 2 is 0.91231 with the standard error 0.10553), but the last two lags are insignificant (the value of lag 14 is 0.00910 with the standard error 0.09504 and the value of lag 15 is 0.05987 with the standard error 0.08779).

Forecasting

The optimal (MMSE) l-step-ahead forecast of $\mathbf{y}_{t+l}$ is

$$\mathbf{y}_{t+l|t} = \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+l-j|t} - \sum_{j=l}^{q} \Theta_j \boldsymbol{\epsilon}_{t+l-j}, \quad l \leq q$$

$$\mathbf{y}_{t+l|t} = \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+l-j|t}, \quad l > q$$

with $\mathbf{y}_{t+l-j|t} = \mathbf{y}_{t+l-j}$ and $\mathbf{x}_{t+l-j|t} = \mathbf{x}_{t+l-j}$ for $l \leq j$. For the forecasts $\mathbf{x}_{t+l-j|t}$, see the State-Space Modeling section.

Covariance Matrices of Prediction Errors without Exogenous (Independent) Variables

Under the stationarity assumption, the optimal (MMSE) l-step-ahead forecast of $\mathbf{y}_{t+l}$ has an infinite moving-average form, $\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$. The prediction error of the optimal l-step-ahead forecast is $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$, with zero mean and covariance matrix

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma \Psi_j' = \sum_{j=0}^{l-1} \Psi_j^o \Psi_j^{o'}$$

where $\Psi_j^o = \Psi_j P$ with a lower triangular matrix P such that $\Sigma = PP'$. Under the assumption of normality of the $\boldsymbol{\epsilon}_t$, the l-step-ahead prediction error $\mathbf{e}_{t+l|t}$ is also normally distributed as multivariate $N(0, \Sigma(l))$. Hence, it follows that the diagonal elements $\sigma_{ii}^2(l)$ of $\Sigma(l)$ can be used, together with the point forecasts $y_{i,t+l|t}$, to construct l-step-ahead prediction interval forecasts of the future values of the component series, $y_{i,t+l}$.

The following statements use the COVPE option to compute the covariance matrices of the prediction errors for a VAR(1) model. The parts of the VARMAX procedure output are shown in Figure 30.31 and Figure 30.32.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(covpe(15))
                 printform=both;
run;
```

```
                              The VARMAX Procedure

                          Prediction Error Covariances

                 Lead    Variable              y1              y2

                    1    y1               1.28875         0.39751
                         y2               0.39751         1.41839
                    2    y1               2.92119         1.00189
                         y2               1.00189         2.18051
                    3    y1               4.59984         1.98771
                         y2               1.98771         3.03498
                    :                       :               :
                   14    y1               7.93640         4.89643
                         y2               4.89643         6.84041
                   15    y1               7.94811         4.90204
                         y2               4.90204         6.86092
```

Figure 30.31. Covariances of Prediction Errors (COVPE Option)

Figure 30.31 is the output in a matrix format associated with the COVPE option for the prediction error covariance matrices.

```
                              The VARMAX Procedure

                   Prediction Error Covariances by Variable

              Variable       Lead              y1              y2

              y1                1         1.28875         0.39751
                                2         2.92119         1.00189
                                3         4.59984         1.98771
                                :           :               :
                               14         7.93640         4.89643
                               15         7.94811         4.90204
              y2                1         0.39751         1.41839
                                2         1.00189         2.18051
                                3         1.98771         3.03498
                                :           :               :
                               14         4.89643         6.84041
                               15         4.90204         6.86092
```

Figure 30.32. Covariances of Prediction Errors Continued

Figure 30.32 is the output in a univariate format associated with the COVPE option for the prediction error covariances. This printing format more easily explains the forecast limit of each variable.

Covariance Matrices of Prediction Errors in the Presence of Exogenous (Independent) Variables

Exogenous variables can be both stochastic and nonstochastic (deterministic) variables. Considering the forecasts in the VARMAX(p,q,s) model, there are two cases.

When exogenous (independent) variables are stochastic (future values not specified)

As defined in the "State-space Modeling" section, $\mathbf{y}_{t+l|t}$ has the representation

$$\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} V_j \mathbf{a}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and hence

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} V_j \mathbf{a}_{t+l-j} + \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

Therefore, the covariance matrix of the l-step-ahead prediction error is given as

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} V_j \Sigma_a V_j' + \sum_{j=0}^{l-1} \Psi_j \Sigma_\epsilon \Psi_j'$$

where Σ_a is the covariance of the white noise series $\mathbf{a}_t$, where $\mathbf{a}_t$ is the white noise series for the VARMA(p,q) model of exogenous (independent) variables, which is assumed not to be correlated with $\boldsymbol{\epsilon}_t$ or its lags.

When future exogenous (independent) variables are specified

The optimal forecast $\mathbf{y}_{t+l|t}$ of $\mathbf{y}_t$ conditioned on the past information and also on known future values $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+l}$ can be represented as

$$\mathbf{y}_{t+l|t} = \sum_{j=0}^{\infty} \Psi_j^* \mathbf{x}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and the forecast error is

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

Thus, the covariance matrix of the l-step-ahead prediction error is given as

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma_\epsilon \Psi_j'$$

Decomposition of Prediction Error Covariances

In the relation $\Sigma(l) = \sum_{j=0}^{l-1} \Psi_j^o \Psi_j^{o\prime}$, the diagonal elements can be interpreted as providing a decomposition of the l-step-ahead prediction error covariance $\sigma_{ii}^2(l)$ for each component series y_{it} into contributions from the components of the standardized innovations $\boldsymbol{\epsilon}_t$.

If you denote the (i,n)th element of Ψ_j^o by $\psi_{j,in}$, the MSE of $y_{i,t+h|t}$ is

$$\mathrm{MSE}(y_{i,t+h|t}) = \mathrm{E}(y_{i,t+h} - y_{i,t+h|t})^2 = \sum_{j=0}^{l-1}\sum_{n=1}^{k} \psi_{j,in}^2$$

Note that $\sum_{j=0}^{l-1} \psi_{j,in}^2$ is interpreted as the contribution of innovations in variable n to the prediction error covariance of the l-step-ahead forecast of variable i.

The proportion, $\omega_{l,in}$, of the l-step-ahead forecast error covariance of variable i accounting for the innovations in variable n is

$$\omega_{l,in} = \sum_{j=0}^{l-1} \psi_{j,in}^2 / \mathrm{MSE}(y_{i,t+h|t})$$

The following statements use the DECOMPOSE option to compute the decomposition of prediction error covariances and their proportions for a VAR(1) model:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(decompose(15))
                 printform=univariate;
run;
```

```
                          The VARMAX Procedure

                     Proportions of Prediction Error
                         Covariances by Variable

            Variable        Lead             y1             y2

            y1                 1        1.00000        0.00000
                               2        0.88436        0.11564
                               3        0.75132        0.24868
                               :          :              :
                              14        0.55184        0.44816
                              15        0.55237        0.44763
            y2                 1        0.08644        0.91356
                               2        0.31767        0.68233
                               3        0.50247        0.49753
                               :          :              :
                              14        0.46611        0.53389
                              15        0.46473        0.53527
```

Figure 30.33. Decomposition of Prediction Error Covariances (DECOMPOSE Option)

The proportions of decomposition of prediction error covariances of two variables are given in Figure 30.33. The output explains that about 91% of the one-step-ahead prediction error covariances of the variable y_{2t} is accounted for by its own innovations and about 9% is accounted for by y_{1t} innovations. For the long-term forecasts, 53.5% and 46.5% of the error variance is accounted for by y_{2t} and y_{1t} innovations.

Forecasting of the Centered Series

If the CENTER option is specified, the sample mean vector is added to the forecast.

Forecasting of the Differenced Series

If dependent (endogenous) variables are differenced, the final forecasts and their prediction error covariances are produced by integrating those of the differenced series. However, if the PRIOR option is specified, the forecasts and their prediction error variances of the differenced series are produced.

Let $\mathbf{z}_t$ be the original series with some zero values appended corresponding to the unobserved past observations. Let $\Delta(B)$ be the $k \times k$ matrix polynomial in the backshift operator corresponding to the differencing specified by the MODEL statement. The off-diagonal elements of Δ_i are zero and the diagonal elements can be different. Then $\mathbf{y}_t = \Delta(B)\mathbf{z}_t$.

This gives the relationship

$$\mathbf{z}_t = \Delta^{-1}(B)\mathbf{y}_t = \sum_{j=0}^{\infty} \Lambda_j \mathbf{y}_{t-j}$$

where $\Delta^{-1}(B) = \sum_{j=0}^{\infty} \Lambda_j B^j$ and $\Lambda_0 = I_k$.

The l-step-ahead prediction of $\mathbf{z}_{t+l}$ is

$$\mathbf{z}_{t+l|t} = \sum_{j=0}^{l-1} \Lambda_j \mathbf{y}_{t+l-j|t} + \sum_{j=l}^{\infty} \Lambda_j \mathbf{y}_{t+l-j}$$

The l-step-ahead prediction error of $\mathbf{z}_{t+l}$ is

$$\sum_{j=0}^{l-1} \Lambda_j \left(\mathbf{y}_{t+l-j} - \mathbf{y}_{t+l-j|t}\right) = \sum_{j=0}^{l-1} \left(\sum_{u=0}^{j} \Lambda_u \Psi_{j-u} \right) \boldsymbol{\epsilon}_{t+l-j}$$

Letting $\Sigma_{\mathbf{z}}(0) = 0$, the covariance matrix of the l-step-ahead prediction error of $\mathbf{z}_{t+l}$, $\Sigma_{\mathbf{z}}(l)$, is

$$\Sigma_{\mathbf{z}}(l) \quad = \quad \sum_{j=0}^{l-1} \left(\sum_{u=0}^{j} \Lambda_u \Psi_{j-u} \right) \Sigma_{\epsilon} \left(\sum_{u=0}^{j} \Lambda_u \Psi_{j-u} \right)'$$

$$= \quad \Sigma_{\mathbf{z}}(l-1) + \left(\sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j}\right) \Sigma_{\epsilon} \left(\sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j}\right)'$$

If there are stochastic exogenous (independent) variables, the covariance matrix of the l-step-ahead prediction error of $\mathbf{z}_{t+l}$, $\Sigma_{\mathbf{z}}(l)$, is

$$\begin{aligned}\Sigma_{\mathbf{z}}(l) \quad &= \quad \Sigma_{\mathbf{z}}(l-1) + \left(\sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j}\right) \Sigma_{\epsilon} \left(\sum_{j=0}^{l-1} \Lambda_j \Psi_{l-1-j}\right)' \\ &\quad + \left(\sum_{j=0}^{l-1} \Lambda_j V_{l-1-j}\right) \Sigma_{a} \left(\sum_{j=0}^{l-1} \Lambda_j V_{l-1-j}\right)'\end{aligned}$$

Tentative Order Selection

Sample Cross-Covariance and Cross-Correlation Matrices

Given a stationary multivariate time series $\mathbf{y}_t$, cross-covariance matrices are

$$\Gamma(l) = \mathrm{E}[(\mathbf{y}_t - \boldsymbol{\mu})(\mathbf{y}_{t+l} - \boldsymbol{\mu})']$$

where $\boldsymbol{\mu} = \mathrm{E}(\mathbf{y}_t)$, and cross-correlation matrices are

$$\rho(l) = D^{-1}\Gamma(l)D^{-1}$$

where D is a diagonal matrix with the standard deviations of the components of $\mathbf{y}_t$ on the diagonal.

The sample cross-covariance matrix at lag l, denoted as $C(l)$, is computed as

$$\hat{\Gamma}(l) = C(l) = \frac{1}{T}\sum_{t=1}^{T-l} \tilde{\mathbf{y}}_t \tilde{\mathbf{y}}'_{t+l}$$

where $\tilde{\mathbf{y}}_t$ is the centered data and T is the number of nonmissing observations. Thus, $\hat{\Gamma}(l)$ has (i,j)th element $\hat{\gamma}_{ij}(l) = c_{ij}(l)$. The sample cross-correlation matrix at lag l is computed as

$$\hat{\rho}_{ij}(l) = c_{ij}(l)/[c_{ii}(0)c_{jj}(0)]^{1/2}, \;\; i,j = 1,\ldots,k$$

The following statements use the CORRY option to compute the sample cross-correlation matrices and their summary indicator plots in terms of $+$, $-$, and $\cdot$:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3 print=(corry)
                 printform=univariate;
run;
```

```
                              The VARMAX Procedure

               Cross Correlations of Dependent Series by Variable

               Variable        Lag              y1              y2

               y1                0         1.00000         0.67041
                                 1         0.83143         0.84330
                                 2         0.56094         0.81972
                                 3         0.26629         0.66154
               y2                0         0.67041         1.00000
                                 1         0.29707         0.77132
                                 2        -0.00936         0.48658
                                 3        -0.22058         0.22014


                           Schematic Representation
                             of Cross Correlations
                        Variable/
                        Lag          0     1     2     3

                        y1           ++    ++    ++    ++
                        y2           ++    ++    .+    -+

                           + is > 2*std error,  - is <
                           -2*std error,  . is between
```

Figure 30.34. Cross-Correlations (CORRY Option)

Figure 30.34 shows the sample cross-correlation matrices of y_{1t} and y_{2t}. As shown, the sample autocorrelation functions for each variable decay quickly, but are significant with respect to two standard errors.

Partial Autoregressive Matrices

For each $m = 1, 2, \ldots,$ you can define a sequence of matrices Φ_{mm}, which is called the partial autoregression matrices of lag m, as the solution for Φ_{mm} to the Yule-Walker equations of order m,

$$\Gamma(l) = \sum_{i=1}^{m} \Gamma(l-i)\Phi'_{im}, \quad l = 1, 2, \ldots, m$$

The sequence of the partial autoregression matrices Φ_{mm} of order m has the characteristic property that if the process follows the AR(p), then $\Phi_{pp} = \Phi_p$ and $\Phi_{mm} = 0$ for $m > p$. Hence, the matrices Φ_{mm} have the cutoff property for a VAR(p) model, and so they can be useful in the identification of the order of a pure VAR model.

The following statements use the PARCOEF option to compute the partial autoregression matrices:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3 print=(parcoef);
run;
```

```
                        The VARMAX Procedure

                      Partial Autoregression

       Lag     Variable             y1              y2

         1     y1              1.14844        -0.50954
               y2              0.54985         0.37409
         2     y1             -0.00724         0.05138
               y2              0.02409         0.05909
         3     y1             -0.02578         0.03885
               y2             -0.03720         0.10149


                    Schematic Representation
                     of Partial Autoregression
                   Variable/
                   Lag           1     2     3

                   y1            +-    ..    ..
                   y2            ++    ..    ..

                   + is > 2*std error,  - is <
                   -2*std error,  . is between
```

Figure 30.35. Partial Autoregression Matrices (PARCOEF Option)

Figure 30.35 shows that the model can be obtained by an AR order $m = 1$ since partial autoregression matrices are insignificant after lag 1 with respect to two standard errors. The matrix for lag 1 is the same as the Yule-Walker autoregressive matrix.

Partial Correlation Matrices

Define the forward autoregression

$$\mathbf{y}_t = \sum_{i=1}^{m-1} \Phi_{i,m-1}\mathbf{y}_{t-i} + \mathbf{u}_{m,t}$$

and the backward autoregression

$$\mathbf{y}_{t-m} = \sum_{i=1}^{m-1} \Phi^*_{i,m-1}\mathbf{y}_{t-m+i} + \mathbf{u}^*_{m,t-m}$$

The matrices $P(m)$ defined by Ansley and Newbold (1979) are given by

$$P(m) = \Sigma^{*1/2}_{m-1}\Phi'_{mm}\Sigma^{-1/2}_{m-1}$$

where

$$\Sigma_{m-1} = \mathrm{Cov}(\mathbf{u}_{m,t}) = \Gamma(0) - \sum_{i=1}^{m-1}\Gamma(-i)\Phi'_{i,m-1}$$

and

$$\Sigma^*_{m-1} = \mathrm{Cov}(\mathbf{u}^*_{m,t-m}) = \Gamma(0) - \sum_{i=1}^{m-1} \Gamma(m-i)\Phi^{*'}_{m-i,m-1}$$

$P(m)$ is called the partial cross-correlations matrices at lag m between the elements of $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, given $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$. The matrices $P(m)$ have the cutoff property for a VAR(p) model, and so they can be useful in the identification of the order of a pure VAR structure.

The following statements use the PCORR option to compute the partial cross-correlations matrices:

```
   proc varmax data=simul1;
      model y1 y2 / p=1 noint lagmax=3 print=(pcorr)
                    printform=univariate;
   run;
```

```
                                The VARMAX Procedure

                    Partial Cross Correlations by Variable

              Variable        Lag            y1             y2

              y1                1       0.80348        0.42672
                                2       0.00276        0.03978
                                3      -0.01091        0.00032
              y2                1      -0.30946        0.71906
                                2       0.04676        0.07045
                                3       0.01993        0.10676


                          Schematic Representation of
                          Partial Cross Correlations
                          Variable/
                          Lag          1     2     3

                          y1           ++    ..    ..
                          y2           -+    ..    ..

                          + is > 2*std error,  - is <
                          -2*std error,  . is between
```

Figure 30.36. Partial Correlations (PCORR Option)

The partial cross-correlation matrices in Figure 30.36 are insignificant after lag 1 with respect to two standard errors. This indicates that an AR order of $m = 1$ can be an appropriate choice.

Partial Canonical Correlation Matrices

The partial canonical correlations at lag m between the vectors $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, given $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$, are $1 \geq \rho_1(m) \geq \rho_2(m) \cdots \geq \rho_k(m)$. The partial canonical correlations are the canonical correlations between the residual series $\mathbf{u}_{m,t}$ and $\mathbf{u}^*_{m,t-m}$, where $\mathbf{u}_{m,t}$ and $\mathbf{u}^*_{m,t-m}$ are defined in the previous section. Thus, the squared partial canonical correlations $\rho_i^2(m)$ are the eigenvalues of the matrix

$$\{\mathrm{Cov}(\mathbf{u}_{m,t})\}^{-1}\mathrm{E}(\mathbf{u}_{m,t}\mathbf{u}^{*'}_{m,t-m})\{\mathrm{Cov}(\mathbf{u}^*_{m,t-m})\}^{-1}\mathrm{E}(\mathbf{u}^*_{m,t-m}\mathbf{u}^{'}_{m,t}) = \Phi^{*'}_{mm}\Phi^{'}_{mm}$$

It follows that the test statistic to test for $\Phi_m = 0$ in the VAR model of order $m > p$ is approximately

$$(T-m)\ \mathrm{tr}\ \{\Phi^{*'}_{mm}\Phi^{'}_{mm}\} \approx (T-m)\sum_{i=1}^{k}\rho_i^2(m)$$

and has an asymptotic chi-square distribution with k^2 degrees of freedom for $m > p$.

The following statements use the PCANCORR option to compute the partial canonical correlations:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3 print=(pcancorr);
run;
```

```
                              The VARMAX Procedure

                        Partial Canonical Correlations

   Lag     Correlation1     Correlation2         DF     Chi-Square     Pr > ChiSq

     1          0.91783          0.77335          4         142.61         <.0001
     2          0.09171          0.01816          4           0.86         0.9307
     3          0.10861          0.01078          4           1.16         0.8854
```

Figure 30.37. Partial Canonical Correlations (PCANCORR Option)

Figure 30.37 shows that the partial canonical correlations $\rho_i(m)$ between $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$ are {0.918, 0.773}, {0.092, 0.018}, and {0.109, 0.011} for lags m =1 to 3. After lag m =1, the partial canonical correlations are insignificant with respect to the 0.05 significance level, indicating that an AR order of $m = 1$ can be an appropriate choice.

The Minimum Information Criterion (MINIC) Method

The **MIN**imum **I**nformation **C**riterion (MINIC) method can tentatively identify the orders of a VARMA(p,q) process. Note that Spliid (1983), Koreisha and Pukkila (1989), and Quinn (1980) proposed this method. The first step of this method is to obtain estimates of the innovations series, $\boldsymbol{\epsilon}_t$, from the VAR(p_ϵ), where p_ϵ is chosen sufficiently large. The choice of the autoregressive order, p_ϵ, is determined by use of a selection criterion. From the selected VAR(p_ϵ) model, you obtain estimates of residual series

$$\tilde{\boldsymbol{\epsilon}}_t = \mathbf{y}_t - \sum_{i=1}^{p_\epsilon} \hat{\Phi}_i^{p_\epsilon} \mathbf{y}_{t-i} - \hat{\boldsymbol{\delta}}^{p_\epsilon}, \;\; t = p_\epsilon + 1, \ldots, T$$

In the second step, you select the order (p, q) of the VARMA model for p in $(p_{min} : p_{max})$ and q in $(q_{min} : q_{max})$

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} - \sum_{i=1}^{q} \Theta_i \tilde{\boldsymbol{\epsilon}}_{t-i} + \boldsymbol{\epsilon}_t$$

which minimizes a selection criterion like SBC or HQ.

The following statements use the MINIC= option to compute a table containing the information criterion associated with various AR and MA orders:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint minic=(p=3 q=3);
run;
```

```
                           The VARMAX Procedure

                      Minimum Information Criterion

       Lag            MA 0           MA 1           MA 2           MA 3

       AR 0      3.2859653      3.0570091      2.7272377      2.3526368
       AR 1      0.4862246      0.6507991      0.7120257      0.7859524
       AR 2      0.5800236      0.7407785       0.802996       0.850487
       AR 3      0.6696452      0.8131261      0.8395558      0.9094856
```

Figure 30.38. MINIC= Option

Figure 30.38 shows the output associated with the MINIC= option. The criterion takes the smallest value at AR order 1.

VAR Modeling

The pth-order VAR process is written as

$$\mathbf{y}_t - \boldsymbol{\mu} = \sum_{i=1}^{p} \Phi_i (\mathbf{y}_{t-i} - \boldsymbol{\mu}) + \boldsymbol{\epsilon}_t \;\; \text{or} \;\; \Phi(B)(\mathbf{y}_t - \boldsymbol{\mu}) = \boldsymbol{\epsilon}_t$$

Equivalently, it can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t \;\; \text{or} \;\; \Phi(B)\mathbf{y}_t = \boldsymbol{\delta} + \boldsymbol{\epsilon}_t$$

with $\boldsymbol{\delta} = (I_k - \sum_{i=1}^{p} \Phi_i)\boldsymbol{\mu}$.

Stationarity

For stationarity of the VAR process, it must be expressible in the convergent causal infinite MA form as

$$\mathbf{y}_t = \boldsymbol{\mu} + \sum_{j=0}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t-j}$$

where $\Psi(B) = \Phi(B)^{-1} = \sum_{j=0}^{\infty} \Psi_j B^j$ with $\sum_{j=0}^{\infty} ||\Psi_j|| < \infty$, where $||A||$ denotes a norm for the matrix A such as $||A||^2 = \mathrm{tr}\{A'A\}$. The matrix Ψ_j can be recursively obtained from the relation $\Phi(B)\Psi(B) = I$, and is

$$\Psi_j = \Phi_1\Psi_{j-1} + \Phi_2\Psi_{j-2} + \cdots + \Phi_p\Psi_{j-p}$$

where $\Psi_0 = I_k$ and $\Psi_j = 0$ for $j < 0$.

The stationarity condition is satisfied if all roots of $|\Phi(z)| = 0$ are outside of the unit circle. The stationarity condition is equivalent to the condition in the corresponding VAR(1) representation, $\mathbf{Y}_t = \Phi\mathbf{Y}_{t-1} + \boldsymbol{\varepsilon}_t$, that all eigenvalues of the $kp \times kp$ companion matrix Φ be less than one in absolute value, where $\mathbf{Y}_t = (\mathbf{y}'_t, \ldots, \mathbf{y}'_{t-p+1})'$, $\boldsymbol{\varepsilon}_t = (\boldsymbol{\epsilon}'_t, 0', \ldots, 0')'$, and

$$\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

If the stationarity condition is not satisfied, a nonstationary model (a differenced model or an error correction model) may be more appropriate.

The following statements estimate a VAR(1) model and use the ROOTS option to compute the characteristic polynomial roots:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(roots);
run;
```

```
                           The VARMAX Procedure

                   Roots of AR Characteristic Polynomial

Index          Real       Imaginary        Modulus         Radian         Degree

    1       0.77238         0.35899         0.8517         0.4351        24.9284
    2       0.77238        -0.35899         0.8517        -0.4351       -24.9284
```

Figure 30.39. Stationarity (ROOTS Option)

Figure 30.39 shows the output associated with the ROOTS option, which indicates that the series is stationary since the modulus of the eigenvalue is less than one.

Parameter Estimation

Consider the stationary VAR(p) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$ are assumed to be available (for convenience of notation). This can be represented by the general form of the multivariate linear model,

$$Y = XB + E \;\text{ or }\; \mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

where

$$\begin{aligned}
Y &= (\mathbf{y}_1, \ldots, \mathbf{y}_T)' \\
B &= (\boldsymbol{\delta}, \Phi_1, \ldots, \Phi_p)' \\
X &= (X_0, \ldots, X_{T-1})' \\
X_t &= (1, \mathbf{y}'_t, \ldots, \mathbf{y}'_{t-p+1})' \\
E &= (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)' \\
\mathbf{y} &= \text{vec}(Y') \\
\boldsymbol{\beta} &= \text{vec}(B') \\
\mathbf{e} &= \text{vec}(E')
\end{aligned}$$

with *vec* denoting the column stacking operator.

The conditional least-squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = ((X'X)^{-1}X' \otimes I_k)\mathbf{y}$$

and the estimate of Σ is

$$\hat{\Sigma} = (T - (kp+1))^{-1} \sum_{t=1}^{T} \hat{\boldsymbol{\epsilon}}_t \hat{\boldsymbol{\epsilon}}_t{}'$$

where $\hat{\boldsymbol{\epsilon}}_t$ is the residual vectors. Consistency and asymptotic normality of the LS estimator are that

$$\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

where $X'X/T$ converges in probability to Γ_p and $\xrightarrow{d}$ denotes convergence in distribution.

The (conditional) maximum likelihood estimator in the VAR(p) model is equal to the (conditional) least-squares estimator on the assumption of normality of the error vectors.

Asymptotic Distributions of Impulse Response Functions

As before, *vec* denotes the column stacking operator and *vech* is the corresponding operator that stacks the elements on and below the diagonal. The commutation matrix K_k defines as $K_k\text{vec}(A) = \text{vec}(A')$; the duplication matrix D_k, $D_k\text{vech}(A) = \text{vec}(A)$; the elimination matrix L_k, $L_k\text{vec}(A) = \text{vech}(A)$, for any $k \times k$ matrix A.

The asymptotic distributions of the impulse response function is

$$\sqrt{T}\text{vec}(\hat{\Psi}_j - \Psi_j) \xrightarrow{d} N(0, G_j \Sigma_{\boldsymbol{\beta}} G_j') \;\; j = 1, 2, \ldots$$

where $\Sigma_{\boldsymbol{\beta}} = \Gamma_p^{-1} \otimes \Sigma$ and

$$G_j = \frac{\partial \text{vec}(\Psi_j)}{\partial \boldsymbol{\beta}'} = \sum_{i=0}^{j-1} \mathbf{J}(\boldsymbol{\Phi}')^{j-1-i} \otimes \Psi_i$$

where $\mathbf{J} = [I_k, 0, \ldots, 0]$ is a $k \times kp$ matrix and $\boldsymbol{\Phi}$ is a $kp \times kp$ companion matrix.

The asymptotic distributions of the accumulated impulse response function is

$$\sqrt{T}\text{vec}(\hat{\Psi}_l^a - \Psi_l^a) \xrightarrow{d} N(0, F_l \Sigma_{\boldsymbol{\beta}} F_l') \;\; l = 1, 2, \ldots$$

where $F_l = \sum_{j=1}^{l} G_j$.

The asymptotic distributions of the orthogonalized impulse response function is

$$\sqrt{T}\text{vec}(\hat{\Psi}_j^o - \Psi_j^o) \xrightarrow{d} N(0, C_j \Sigma_{\boldsymbol{\beta}} C_j' + \bar{C}_j \Sigma_{\boldsymbol{\sigma}} \bar{C}_j') \;\; j = 0, 1, 2, \ldots$$

where $C_0 = 0$, $C_j = (\Psi_0^{o'} \otimes I_k)G_j$, $\bar{C}_j = (I_k \otimes \Psi_j)H$ and

$$H = \frac{\partial \text{vec}(\Psi_0^o)}{\partial \boldsymbol{\sigma}'} = L_k'\{L_k(I_{k^2} + K_k)(\Psi_0^o \otimes I_k)L_k'\}^{-1}$$

and $\Sigma_{\boldsymbol{\sigma}} = 2D_k^+(\Sigma \otimes \Sigma)D_k^{+'}$ with $D_k^+ = (D_k' D_k)^{-1} D_k'$ and $\boldsymbol{\sigma} = \text{vech}(\Sigma)$.

Granger-Causality Test

Let $\mathbf{y}_t$ be arranged and partitioned in subgroups $\mathbf{y}_{1t}$ and $\mathbf{y}_{2t}$ with dimensions k_1 and k_2, respectively ($k = k_1 + k_2$); that is, $\mathbf{y}_t = (\mathbf{y}_{1t}', \mathbf{y}_{2t}')'$ with the corresponding white noise process $\boldsymbol{\epsilon}_t = (\boldsymbol{\epsilon}_{1t}', \boldsymbol{\epsilon}_{2t}')'$. Consider the VAR($p$) model with partitioned coefficients $\Phi_{ij}(B)$ for $i, j = 1, 2$ as follows:

$$\begin{bmatrix} \Phi_{11}(B) & \Phi_{12}(B) \\ \Phi_{21}(B) & \Phi_{22}(B) \end{bmatrix} \begin{bmatrix} \mathbf{y}_{1t} \\ \mathbf{y}_{2t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

The variables $\mathbf{y}_{1t}$ are said to cause $\mathbf{y}_{2t}$, but $\mathbf{y}_{2t}$ do not cause $\mathbf{y}_{1t}$ if $\Phi_{12}(B) = 0$. The implication of this model structure is that future values of the process $\mathbf{y}_{1t}$ are influenced only by its own past and not by the past of $\mathbf{y}_{2t}$, where future values of $\mathbf{y}_{2t}$ are influenced by the past of both $\mathbf{y}_{1t}$ and $\mathbf{y}_{2t}$. If the future $\mathbf{y}_{1t}$ are not influenced by the past values of $\mathbf{y}_{2t}$, then it can be better to model $\mathbf{y}_{1t}$ separately from $\mathbf{y}_{2t}$.

Consider testing $H_0\colon C\boldsymbol{\beta} = c$, where C is a $s \times (k^2p + k)$ matrix of rank s and c is a s-dimensional vector where $s = k_1k_2p$. Assuming that

$$\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

you get the Wald statistic

$$T(C\hat{\boldsymbol{\beta}} - c)'[C(\hat{\Gamma}_p^{-1} \otimes \hat{\Sigma})C']^{-1}(C\hat{\boldsymbol{\beta}} - c) \xrightarrow{d} \chi^2(s)$$

For the Granger-Causality Test, the matrix C consists of zeros or ones and c is the zero vector.

VARX Modeling

The Vector AutoRegressive model with eXogenous variables is called the VARX(p,s) model. The form of the VARX(p,s) model can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

The parameter estimates can be obtained by representing the general form of the multivariate linear model,

$$Y = XB + E \ \text{ or } \ \mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

where

$$\begin{aligned}
Y &= (\mathbf{y}_1, \ldots, \mathbf{y}_T)' \\
B &= (\boldsymbol{\delta}, \Phi_1, \ldots, \Phi_p, \Theta_0^*, \ldots, \Theta_s^*)' \\
X &= (X_0, \ldots, X_{T-1})' \\
X_t &= (1, \mathbf{y}_t', \ldots, \mathbf{y}_{t-p+1}', \mathbf{x}_{t+1}', \ldots, \mathbf{x}_{t-s+1}')' \\
E &= (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)' \\
\mathbf{y} &= \text{vec}(Y') \\
\boldsymbol{\beta} &= \text{vec}(B') \\
\mathbf{e} &= \text{vec}(E')
\end{aligned}$$

The conditional least-squares estimator of β can be obtained using the same method in a VAR(p) modeling. If the multivariate linear model has different independent variables corresponding to dependent variables, the SUR (Seemingly Unrelated Regression) method is used to improve the regression estimates.

The following example fits the ordinary regression model:

```
proc varmax data=one;
   model y1-y3 = x1-x5;
run;
```

This is equivalent to the REG procedure in the SAS/STAT software.

```
proc reg data=one;
   model y1 = x1-x5;
   model y2 = x1-x5;
   model y3 = x1-x5;
run;
```

The following example fits the second-order lagged regression model:

```
proc varmax data=two;
   model y1 y2 = x / xlag=2;
run;
```

This is equivalent to the REG procedure in the SAS/STAT software.

```
data three;
   set two;
   xlag1 = lag1(x);
   xlag2 = lag2(x);
run;

proc reg data=three;
   model y1 = x xlag1 xlag2;
   model y2 = x xlag1 xlag2;
run;
```

The following example fits the ordinary regression model with different regressors:

```
proc varmax data=one;
   model y1 = x1-x3, y2 = x2 x3;
run;
```

This is equivalent to the following SYSLIN procedure statements.

```
proc syslin data=one vardef=nf sur;
   endogenous y1 y2;
   model y1 = x1-x3;
   model y2 = x2 x3;
run;
```

From the output in Figure 30.20 on page 1714, you can see that the parameters, XL0_1_2, XL0_2_1, XL0_3_1, and XL0_3_2, are not significant. The following example fits the VARX(1,0) model with different regressors:

```
proc varmax data=grunfeld;
   model y1 = x1, y2 = x2, y3 / p=1;
run;
```

```
                    The VARMAX Procedure

          Coefficient Estimates of Independent Variables

           Lag     Variable              x1              x2

             0     y1               1.83231               _
                   y2                     _         2.42110
                   y3                     _               _
```

Figure 30.40. Parameter Estimates for the VARX(1, 0) Model

As you can see in Figure 30.40, the symbol '_' in the elements of matrix correspond to endogenous variables that do not take exogenous variables.

Model Diagnostic Checks

Multivariate Model Diagnostic Checks

- Information Criterion

  Various model selection criteria (normalized by T) can be used to choose the appropriate model. The following list includes the Akaike Information Criterion (AIC), the corrected Akaike Information Criterion (AICC), the Final Prediction Error criterion (FPE), the Hannan-Quinn Criterion (HQC), and the Schwarz Bayesian Criterion (SBC), also referred to as BIC.

$$\begin{aligned}
\text{AIC} &= \log(|\tilde{\Sigma}|) + 2r/T \\
\text{AICC} &= \log(|\tilde{\Sigma}|) + 2r/(T - r/k) \\
\text{FPE} &= (\frac{T + r/k}{T - r/k})^k |\tilde{\Sigma}| \\
\text{HQC} &= \log(|\tilde{\Sigma}|) + 2r\log(\log(T))/T \\
\text{SBC} &= \log(|\tilde{\Sigma}|) + r\log(T)/T
\end{aligned}$$

  where r denotes the number of parameters estimated and $\tilde{\Sigma}$ is the maximum likelihood estimate of Σ.

  An example of the output was displayed in Figure 30.4 on page 1699.

- Portmanteau Statistic Q_s

  Let $C_\epsilon(l)$ be the residual cross-covariance matrices and $\hat{\rho}_\epsilon(l)$ be the residual cross-correlation matrices as

$$C_\epsilon(l) = T^{-1}\sum_{t=1}^{T-l} \boldsymbol{\epsilon}_t \boldsymbol{\epsilon}'_{t+l}$$

and

$$\hat{\rho}_\epsilon(l) = \hat{V}_\epsilon^{-1/2} C_\epsilon(l) \hat{V}_\epsilon^{-1/2} \text{ and } \hat{\rho}_\epsilon(-l) = \hat{\rho}_\epsilon(l)'$$

where $\hat{V}_\epsilon = \mathrm{Diag}(\hat{\sigma}_{11}^2, \ldots, \hat{\sigma}_{kk}^2)$ and $\hat{\sigma}_{ii}^2$ are the diagonal elements of $\hat{\Sigma}$. The multivariate portmanteau test defined in Hosking (1980) is

$$Q_s = T^2 \sum_{l=1}^{s} (T-l)^{-1} \mathrm{tr}\{\hat{\rho}_\epsilon(l) \Sigma^{-1} \hat{\rho}_\epsilon(-l) \Sigma^{-1}\}$$

The statistic Q_s has approximately the chi-square distribution with $k^2(s-p-q)$ degrees of freedom. An example of the output was displayed in Figure 30.7 on page 1700.

Univariate Model Diagnostic Checks

There are various ways to perform diagnostic checks for a univariate model. For details, see the chapter on the ARIMA or AUTOREG procedure. An example of the output was displayed in Figure 30.8 and Figure 30.9 on page 1701.

- Durbin-Watson (DW) statistics: The test statistics are computed from the residuals of the autoregressive model with order 1.
- F tests for autoregressive conditional heteroscedastic (ARCH) disturbances: These test statistics are computed from the residuals of the ARCH(1) model.
- F tests for AR disturbance: These test statistics are computed from the residuals of the univariate AR(1), AR(2), AR(3), and AR(4) models.
- Jarque-Bera normality test: This test is helpful in determining whether the model residuals represent a white noise process.

Bayesian VAR Modeling

Consider the VAR(p) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \cdots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

or

$$\mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

When the parameter vector $\boldsymbol{\beta}$ has a prior multivariate normal distribution with known mean $\boldsymbol{\beta}^*$ and covariance matrix V_β, the prior density is written as

$$f(\boldsymbol{\beta}) = (\frac{1}{2\pi})^{k^2p/2} |V_\beta|^{-1/2} \exp[-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*) V_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}^*)]$$

The likelihood function for the Gaussian process becomes

$$\begin{aligned}\ell(\boldsymbol{\beta}|\mathbf{y}) &= (\frac{1}{2\pi})^{kT/2}|I_T \otimes \Sigma|^{-1/2} \times \\ &\quad \exp[-\frac{1}{2}(\mathbf{y}-(X\otimes I_k)\boldsymbol{\beta})'(I_T\otimes\Sigma^{-1})(\mathbf{y}-(X\otimes I_k)\boldsymbol{\beta})]\end{aligned}$$

Therefore, the posterior density is derived as

$$f(\boldsymbol{\beta}|\mathbf{y}) \propto \exp[-\frac{1}{2}(\boldsymbol{\beta}-\bar{\boldsymbol{\beta}})'\bar{\Sigma}_{\beta}^{-1}(\boldsymbol{\beta}-\bar{\boldsymbol{\beta}})]$$

where the posterior mean is

$$\bar{\boldsymbol{\beta}} = [V_{\beta}^{-1} + (X'X\otimes\Sigma^{-1})]^{-1}[V_{\beta}^{-1}\boldsymbol{\beta}^{*} + (X'\otimes\Sigma^{-1})\mathbf{y}]$$

and the posterior covariance matrix is

$$\bar{\Sigma}_{\beta} = [V_{\beta}^{-1} + (X'X\otimes\Sigma^{-1})]^{-1}$$

In practice, the prior mean $\boldsymbol{\beta}^{*}$ and the prior variance V_{β} need to be specified. If all the parameters are considered to shrink toward zero, the null prior mean should be specified. According to Litterman (1986), the prior variance can be given by

$$v_{ij}(l) = \begin{cases} (\lambda/l)^2 & \text{if } i=j \\ (\lambda\theta\sigma_{ii}/l\sigma_{jj})^2 & \text{if } i\neq j \end{cases}$$

where $v_{ij}(l)$ is the prior variance of the (i,j)th element of Φ_l, λ is the prior standard deviation of the diagonal elements of Φ_l, θ is a constant in the interval $(0,1)$, and σ_{ii}^2 is the ith diagonal element of Σ. The deterministic terms have diffused prior variance. In practice, you replace the σ_{ii}^2 by the diagonal element of the ML estimator of Σ in the nonconstrained model.

For example, for a bivariate BVAR(2) model,

$$\begin{aligned} y_{1t} &= 0 + \phi_{1,11}y_{1,t-1} + \phi_{1,12}y_{2,t-1} + \phi_{2,11}y_{1,t-2} + \phi_{2,12}y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + \phi_{1,21}y_{1,t-1} + \phi_{1,22}y_{2,t-1} + \phi_{2,21}y_{1,t-2} + \phi_{2,22}y_{2,t-2} + \epsilon_{2t} \end{aligned}$$

with the prior covariance matrix

$$\begin{aligned} V_{\beta} = \text{Diag} \quad (& \infty, \lambda^2, (\lambda\theta\sigma_1/\sigma_2)^2, (\lambda/2)^2, (\lambda\theta\sigma_1/2\sigma_2)^2, \\ & \infty, (\lambda\theta\sigma_2/\sigma_1)^2, \lambda^2, (\lambda\theta\sigma_2/2\sigma_1)^2, (\lambda/2)^2 \ ) \end{aligned}$$

For the Bayesian Estimation of integrated systems, the prior mean is set to the first lag of each variable equal to one in its own equation and all other coefficients at zero. For example, for a bivariate BVAR(2) model,

$$\begin{aligned} y_{1t} &= 0 + 1\ y_{1,t-1} + 0\ y_{2,t-1} + 0\ y_{1,t-2} + 0\ y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + 0\ y_{1,t-1} + 1\ y_{2,t-1} + 0\ y_{1,t-2} + 0\ y_{2,t-2} + \epsilon_{2t} \end{aligned}$$

Forecasting of BVAR Modeling

The bootstrap procedure is used to estimate standard errors of the forecast (Litterman 1986). NREP=*B* simulations are performed. In each simulation the following steps are taken:

1. The procedure generates the available number of observations, T, and uniform random integers I_t, where $t = 1, \ldots T$.
2. A new observation, $\tilde{\mathbf{y}}_t$, is obtained as a sum of the forecast based on the estimates coefficients plus the vector of residuals from the I_t; that is,

$$\tilde{\mathbf{y}}_t = \sum_{j=1}^{p} \hat{\Phi}_j \mathbf{y}_{t-j} + \hat{\boldsymbol{\epsilon}}_{I_t}$$

3. A new BVAR model is estimated by using the most recent observations, and a prediction value is made of the most recent observations.

The MSE measure of the l-step-ahead forecast is

$$MSE(l) = \frac{1}{B} \sum_{i=1}^{B} (\tilde{\mathbf{y}}^i_{t+l|t} - \bar{\tilde{\mathbf{y}}}_t)^2$$

where $\bar{\tilde{\mathbf{y}}}_t = (1/B) \sum_{i=1}^{B} \bar{\tilde{\mathbf{y}}}^i_t$.

VARMA Modeling

A VARMA(p, q) process is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

or

$$\Phi(B)\mathbf{y}_t = \boldsymbol{\delta} + \Theta(B)\boldsymbol{\epsilon}_t$$

where $\Phi(B) = I_k - \sum_{i=1}^{p} \Phi_i B^i$ and $\Theta(B) = I_k - \sum_{i=1}^{q} \Theta_i B^i$.

Stationarity and Invertibility

For stationarity and invertibility of the VARMA process, the roots of $|\Phi(z)| = 0$ and $|\Theta(z)| = 0$ are outside the unit circle.

Parameter Estimation

Under the assumption of normality of the $\boldsymbol{\epsilon}_t$ with mean vector zero and nonsingular covariance matrix Σ, consider the conditional (approximate) log-likelihood function of a VARMA(p,q) model with mean zero.

Define $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_T)'$ and $E = (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)'$ with $B^iY = (\mathbf{y}_{1-i}, \ldots, \mathbf{y}_{T-i})'$ and $B^iE = (\boldsymbol{\epsilon}_{1-i}, \ldots, \boldsymbol{\epsilon}_{T-i})'$; define $\mathbf{y} = \mathrm{vec}(Y')$ and $\mathbf{e} = \mathrm{vec}(E')$. Then

$$\mathbf{y} - \sum_{i=1}^{p}(I_T \otimes \Phi_i)B^i\mathbf{y} = \mathbf{e} - \sum_{i=1}^{q}(I_T \otimes \Theta_i)B^i\mathbf{e}$$

where $B^i\mathbf{y} = \mathrm{vec}[(B^iY)']$ and $B^i\mathbf{e} = \mathrm{vec}[(B^iE)']$.

Then, the conditional (approximate) log-likelihood function can be written as the following (see Reinsel 1997):

$$\begin{aligned} \ell &= -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_{t=1}^{T}\boldsymbol{\epsilon}_t'\Sigma^{-1}\boldsymbol{\epsilon}_t \\ &= -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\mathbf{w}'\Theta'^{-1}(I_T \otimes \Sigma^{-1})\Theta^{-1}\mathbf{w} \end{aligned}$$

where $\mathbf{w} = \mathbf{y} - \sum_{i=1}^{p}(I_T \otimes \Phi_i)B^i\mathbf{y}$; Θ such that $\mathbf{e} - \sum_{i=1}^{q}(I_T \otimes \Theta_i)B^i\mathbf{e} = \Theta\mathbf{e}$.

For the exact log-likelihood function of a VARMA(p,q) model, the Kalman filtering method is used transforming the VARMA process into the state-space form (see Reinsel 1997).

The state-space form of the VARMA(p,q) model consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where for $v = \max(p, q+1)$

$$\mathbf{z}_t = (\mathbf{y}_t', \mathbf{y}_{t+1|t}', \ldots, \mathbf{y}_{t+v-1|t}')'$$

$$F = \begin{bmatrix} 0 & I_k & 0 & \cdots & 0 \\ 0 & 0 & I_k & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi_v & \Phi_{v-1} & \Phi_{v-2} & \cdots & \Phi_1 \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ \Psi_1 \\ \vdots \\ \Psi_{v-1} \end{bmatrix}$$

and

$$H = [I_k, 0, \ldots, 0]$$

The Kalman filtering approach is used for evaluation of the likelihood function. The updating equation is

$$\hat{\mathbf{z}}_{t|t} = \hat{\mathbf{z}}_{t|t-1} + K_t \boldsymbol{\epsilon}_{t|t-1}$$

with

$$K_t = P_{t|t-1} H' [H P_{t|t-1} H']^{-1}$$

and the prediction equation is

$$\hat{\mathbf{z}}_{t|t-1} = F \hat{\mathbf{z}}_{t-1|t-1}, \;\; P_{t|t-1} = F P_{t-1|t-1} F' + G \Sigma G'$$

with $P_{t|t} = [I - K_t H] P_{t|t-1}$ for $t = 1, 2, \ldots, n$.

The log-likelihood function can be expressed as

$$\ell = -\frac{1}{2} \sum_{t=1}^{T} [\log |\Sigma_{t|t-1}| - (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})' \Sigma_{t|t-1}^{-1} (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})]$$

where $\hat{\mathbf{y}}_{t|t-1}$ and $\Sigma_{t|t-1}$ determined recursively from the Kalman filter procedure. From Kalman filtering, to construct the likelihood function you obtain $\hat{\mathbf{y}}_{t|t-1} = H \hat{\mathbf{z}}_{t|t-1}$, $\hat{\boldsymbol{\epsilon}}_{t|t-1} = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$, and $\Sigma_{t|t-1} = H P_{t|t-1} H'$.

Define the vector $\boldsymbol{\beta}$

$$\boldsymbol{\beta} = (\phi_1', \ldots, \phi_p', \theta_1', \ldots, \theta_q', \mathrm{vech}(\Sigma))'$$

where $\phi_i = \mathrm{vec}(\Phi_i)$ and $\theta_i = \mathrm{vec}(\Theta_i)$.

The log-likelihood equations are solved by iterative numerical procedures such as the quasi-Newton optimization. The starting values for the AR and MA parameters are obtained from the least squares estimates.

Asymptotic Distribution of the Parameter Estimates

Under the assumptions of stationarity and invertibility for the VARMA model, and the assumption that $\boldsymbol{\epsilon}_t$ is a white noise process, $\hat{\boldsymbol{\beta}}$ is a consistent estimator for $\boldsymbol{\beta}$ and $\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$ converges in distribution to the multivariate normal $N(0, V^{-1})$ as $T \to \infty$, where V is the asymptotic information matrix of $\boldsymbol{\beta}$.

Asymptotic Distributions of Impulse Response Functions

Defining the vector $\boldsymbol{\beta}$

$$\boldsymbol{\beta} = (\phi_1', \ldots, \phi_p', \theta_1', \ldots, \theta_q')'$$

the asymptotic distribution of the impulse response function for a VARMA(p, q) model is

$$\sqrt{T}\mathrm{vec}(\hat{\Psi}_j - \Psi_j) \xrightarrow{d} N(0, G_j \Sigma_{\boldsymbol{\beta}} G_j') \;\; j = 1, 2, \ldots$$

where $\Sigma_{\boldsymbol{\beta}}$ is the covariance matrix of the parameter estimates and

$$G_j = \frac{\partial \mathrm{vec}(\Psi_j)}{\partial \boldsymbol{\beta}'} = \sum_{i=0}^{j-1} \mathbf{H}'(\mathbf{A}')^{j-1-i} \otimes \mathbf{J}\mathbf{A}^i\mathbf{J}'$$

where $\mathbf{H} = [I_k, 0, \ldots, 0, I_k, 0, \ldots, 0]'$ is a $k(p+q) \times k$ matrix with the second I_k follows after p block matrices; $\mathbf{J} = [I_k, 0, \ldots, 0]$ is a $k \times k(p+q)$ matrix; $\mathbf{A}$ is a $k(p+q) \times k(p+q)$ matrix that

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where

$$A_{11} = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix} \quad A_{12} = \begin{bmatrix} -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix}$$

A_{21} is a $kq \times kp$ zero matrix, and

$$A_{22} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

An Example of a VARMA(1,1) Model

Consider a VARMA(1,1) model with mean zero

$$\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t - \Theta_1 \boldsymbol{\epsilon}_{t-1}$$

where $\boldsymbol{\epsilon}_t$ is the white noise process with a mean zero vector and the positive-definite covariance matrix Σ.

The following IML procedure statements simulate a bivariate vector time series from this model to provide test data for the VARMAX procedure:

```
proc iml;
   sig = {1.0  0.5, 0.5 1.25};
   phi = {1.2 -0.5, 0.6 0.3};
   theta = {0.5 -0.2, 0.1 0.3};
   /* to simulate the vector time series */
   call varmasim(y,phi,theta) sigma=sig n=100 seed=34657;
   cn = {'y1' 'y2'};
   create simul3 from y[colname=cn];
   append from y;
quit;
```

The following statements fit a VARMA(1,1) model to the simulated data. You specify the order of the autoregressive model with the P= option and the order of moving-average model with the Q= option. You specify the quasi-Newton optimization in the NLOPTIONS statement as an optimization method.

```
proc varmax data=simul3;
   nloptions tech=qn;
   model y1 y2 / p=1 q=1 noint;
run;
```

```
                     The VARMAX Procedure

                      Optimization Start
                      Parameter Estimates
                                                 Gradient
                                                Objective
               N Parameter         Estimate      Function

               1 AR1_1_1           1.013118     -1.026092
               2 AR1_2_1           0.510233      0.217500
               3 AR1_1_2          -0.399051      0.722673
               4 AR1_2_2           0.441344     -9.015868
               5 MA1_1_1           0.295872     -1.867938
               6 MA1_2_1          -0.002809      2.221207
               7 MA1_1_2          -0.044216     -0.641937
               8 MA1_2_2           0.425334      0.850316
               9 SIGH1_1           1.122565      0.452830
              10 SIGH1_2           0.343605      0.113495
              11 SIGH2_2           1.137912      5.134866
```

Figure 30.41. Start Parameter Estimates for the VARMA(1, 1) Model

Figure 30.41 shows the initial values of parameters. The initial values were estimated using the least squares method. The parameters SIGHi_j ($1 \leq i \leq j \leq 2$) refer to the root of Σ.

```
                               The VARMAX Procedure

               Minimum Iterations                                    0
               Maximum Iterations                                  200
               Maximum Function Calls                             2000
               ABSGCONV Gradient Criterion                     0.00001
               GCONV Gradient Criterion                           1E-8
               ABSFCONV Function Criterion                           0
               FCONV Function Criterion                   2.220446E-16
               FCONV2 Function Criterion                             0
               FSIZE Parameter                                       0
               ABSXCONV Parameter Change Criterion                   0
               XCONV Parameter Change Criterion                      0
               XSIZE Parameter                                       0
               ABSCONV Function Criterion                 -1.34078E154
               Line Search Method                                    2
               Starting Alpha for Line Search                        1
               Line Search Precision LSPRECISION                   0.4
               DAMPSTEP Parameter for Line Search                    .
               Singularity Tolerance (SINGULAR)                   1E-8
```

Figure 30.42. Default Criteria for the Quasi-Newton Optimization

Figure 30.42 shows the default options for the quasi-Newton optimization technique.

```
                               The VARMAX Procedure

                                                            Max Abs             Slope
         Rest     Func        Act     Objective   Obj Fun  Gradient      Step    Search
Iter     arts    Calls        Con      Function    Change   Element      Size    Direc

   1        0       48          0     121.80390    0.1506    7.0747   0.00446   -66.023
   2        0       71          0     121.65001    0.1539    4.5560     1.000    -0.252
   3        0      117          0     121.50360    0.1464    4.7057     2.000    -0.165
   4        0      163          0     121.29783    0.2058    5.9621     3.201    -0.131
   5        0      209          0     121.11225    0.1856    2.3765     2.426    -0.157
   6        0      233          0     121.09059    0.0217    2.4115     1.556   -0.0360
   7        0      256          0     121.06681    0.0238    2.3547     2.066   -0.0262
   8        0      280          0     121.05711   0.00970    1.0649     1.603   -0.0119
   9        0      303          0     121.04549    0.0116    1.1019     3.701   -0.0066
  10        0      327          0     121.04034   0.00515    0.5685     1.635   -0.0061
  11        0      350          0     121.03851   0.00183    0.6229     6.308   -0.0020
  12        0      373          0     121.03557   0.00294    0.1935     0.814   -0.0056
  13        0      397          0     121.03519  0.000378    0.0385     1.058   -0.0007
  14        0      421          0     121.03518  0.000014    0.0105     1.539   -187E-7
  15        0      445          0     121.03518  1.513E-6   0.00324     1.770   -171E-8
  16        0      469          0     121.03518  4.686E-8  0.000601     1.005    -93E-9
```

Figure 30.43. Iteration History of Parameter Estimates

Figure 30.43 shows the iteration history of parameter estimates.

```
                    The VARMAX Procedure

                    Optimization Results
                     Parameter Estimates
                                                  Gradient
                                                 Objective
           N Parameter         Estimate           Function

           1 AR1_1_1           1.018467     -0.000015043
           2 AR1_2_1           0.391796        -0.000161
           3 AR1_1_2          -0.386811        -0.000187
           4 AR1_2_2           0.552816         0.000389
           5 MA1_1_1           0.322920     -0.000028166
           6 MA1_2_1          -0.165034         0.000138
           7 MA1_1_2          -0.021598      0.000063347
           8 MA1_2_2           0.585787        -0.000193
           9 SIGH1_1           1.118936         0.000601
          10 SIGH1_2           0.339197         0.000394
          11 SIGH2_2           1.094569         0.000166
```

Figure 30.44. Results of Parameter Estimates for the VARMA(1, 1) Model

Figure 30.44 shows the final parameter estimates.

```
                         The VARMAX Procedure

               Type of Model                                 VARMA(1,1)
               Estimation Method    Maximum Likelihood Estimation


                       AR Coefficient Estimates

                  Lag    Variable              y1              y2

                    1    y1               1.01847        -0.38681
                         y2               0.39180         0.55282


                       MA Coefficient Estimates

                  Lag    Variable              e1              e2

                    1    y1               0.32292        -0.02160
                         y2              -0.16503         0.58579


                              Schematic
                          Representation of
                         Parameter Estimates

                       Variable/
                       Lag          AR1    MA1

                       y1           +-     +.
                       y2           ++     .+

                       + is > 2*std error,  -
                        is < -2*std error,  .
                        is between,  * is N/A


                      Model Parameter Estimates

                                            Standard
Equation Parameter      Estimate        Error t Value Pr > |t| Variable

y1       AR1_1_1         1.01847      0.10256    9.93   0.0001 y1(t-1)
         AR1_1_2        -0.38681      0.09645   -4.01   0.0001 y2(t-1)
         MA1_1_1         0.32292      0.14525    2.22   0.0285 e1(t-1)
         MA1_1_2        -0.02160      0.14204   -0.15   0.8795 e2(t-1)
y2       AR1_2_1         0.39180      0.10062    3.89   0.0002 y1(t-1)
         AR1_2_2         0.55282      0.08422    6.56   0.0001 y2(t-1)
         MA1_2_1        -0.16503      0.15705   -1.05   0.2959 e1(t-1)
         MA1_2_2         0.58579      0.14115    4.15   0.0001 e2(t-1)
```

Figure 30.45. Parameter Estimates for the VARMA(1, 1) Model

Figure 30.45 shows the AR coefficient matrix in terms of lag 1, the MA coefficient matrix in terms of lag 1, the parameter estimates, and their significance that indicates how well the model fits the data.

The fitted VARMA(1,1) model with estimated standard errors in parentheses are given as

$$\mathbf{y}_t = \begin{pmatrix} 1.018 & -0.387 \\ (0.045) & (0.057) \\ 0.392 & 0.553 \\ (0.043) & (0.053) \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t - \begin{pmatrix} 0.323 & -0.022 \\ (0.120) & (0.126) \\ -0.165 & 0.586 \\ (0.115) & (0.144) \end{pmatrix} \boldsymbol{\epsilon}_{t-1}$$

Cointegration

This section briefly introduces the concepts of cointegration.

Definition 1. *(Engle and Granger 1987): If a series y_t with no deterministic components can be represented by a stationary and invertible ARMA process after differencing d times, the series is integrated of order d, that is, $y_t \sim I(d)$.*

Definition 2. *(Engle and Granger 1987): If all elements of the vector $\mathbf{y}_t$ are $I(d)$ and there exists a cointegrating vector $\boldsymbol{\beta} \neq 0$ such that $\boldsymbol{\beta}'\mathbf{y}_t \sim I(d-b)$ for any $b > 0$, the vector process is said to be cointegrated $CI(d, b)$.*

A simple example of a cointegrated process is the following bivariate system:

$$\begin{aligned} y_{1t} &= \gamma y_{2t} + \epsilon_{1t} \\ y_{2t} &= y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

with ϵ_{1t} and ϵ_{2t} being uncorrelated white noise processes. In the second equation, y_{2t} is a random walk, $\Delta y_{2t} = \epsilon_{2t}$, $\Delta \equiv 1 - B$. Differencing the first equation results in

$$\Delta y_{1t} = \gamma \Delta y_{2t} + \Delta \epsilon_{1t} = \gamma \epsilon_{2t} + \epsilon_{1t} - \epsilon_{1,t-1}$$

Thus, both y_{1t} and y_{2t} are $I(1)$ processes, but the linear combination $y_{1t} - \gamma y_{2t}$ is stationary. Hence $\mathbf{y}_t = (y_{1t}, y_{2t})'$ is cointegrated with a cointegrating vector $\boldsymbol{\beta} = (1, -\gamma)'$.

In general, if the vector process $\mathbf{y}_t$ has k components, then there can be more than one cointegrating vector $\boldsymbol{\beta}'$. It is assumed that there are r linearly independent cointegrating vectors with $r < k$, which make the $k \times r$ matrix $\boldsymbol{\beta}$. The rank of matrix $\boldsymbol{\beta}$ is r, which is called the *cointegration rank* of $\mathbf{y}_t$.

Common Trends

This section briefly discusses the implication of cointegration for the moving-average representation. Let $\mathbf{y}_t$ be cointegrated $CI(1, 1)$, then $\Delta\mathbf{y}_t$ has the Wold representation.

$$\Delta\mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t, \ \sum_{j=0}^{\infty} j|\Psi_j| < \infty$$

where $\boldsymbol{\epsilon}_t$ is $iid(0, \Sigma)$ and $\Psi(B) = \sum_{j=0}^{\infty} \Psi_j B^j$ with $\Psi_0 = I_k$.

Assume that $\boldsymbol{\epsilon}_t = 0$ if $t \leq 0$ and $\mathbf{y}_0$ is a nonrandom initial value. Then the difference equation implies that

$$\mathbf{y}_t = \mathbf{y}_0 + \boldsymbol{\delta} t + \Psi(1) \sum_{i=0}^{t} \boldsymbol{\epsilon}_i + \Psi^*(B)\boldsymbol{\epsilon}_t$$

where $\Psi^*(B) = (1-B)^{-1}(\Psi(B) - \Psi(1))$ and $\Psi^*(B)$ is absolutely summable.

Assume that the rank of $\Psi(1)$ is $m = k - r$. When the process $\mathbf{y}_t$ is cointegrated, there is a cointegrating $k \times r$ matrix $\boldsymbol{\beta}$ such that $\boldsymbol{\beta}'\mathbf{y}_t$ is stationary.

Premultiplying $\mathbf{y}_t$ by $\boldsymbol{\beta}'$ results in

$$\boldsymbol{\beta}'\mathbf{y}_t = \boldsymbol{\beta}'\mathbf{y}_0 + \boldsymbol{\beta}'\Psi^*(B)\boldsymbol{\epsilon}_t$$

because $\boldsymbol{\beta}'\Psi(1) = 0$ and $\boldsymbol{\beta}'\boldsymbol{\delta} = 0$.

Stock and Watson (1988) showed that the cointegrated process $\mathbf{y}_t$ has a common trends representation derived from the moving-average representation. Since the rank of $\Psi(1)$ is $m = k - r$, there is a $k \times r$ matrix H_1 with rank r such that $\Psi(1)H_1 = 0$. Let H_2 be a $k \times m$ matrix with rank m such that $H_2' H_1 = 0$, then $A = C(1)H_2$ has rank m. The $H = (H_1, H_2)$ has rank k. By construction of H,

$$\Psi(1)H = [0, A] = AS_m$$

where $S_m = (0_{m \times r}, I_m)$. Since $\boldsymbol{\beta}'\Psi(1) = 0$ and $\boldsymbol{\beta}'\boldsymbol{\delta} = 0$, $\boldsymbol{\delta}$ lies in the column space of $\Psi(1)$ and can be written

$$\boldsymbol{\delta} = C(1)\tilde{\boldsymbol{\delta}}$$

where $\tilde{\boldsymbol{\delta}}$ is a k-dimensional vector. The common trends representation is written as

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{y}_0 + \Psi(1)[\tilde{\boldsymbol{\delta}} t + \sum_{i=0}^{t} \boldsymbol{\epsilon}_i] + \Psi^*(B)\boldsymbol{\epsilon}_t \\
&= \mathbf{y}_0 + \Psi(1)H[H^{-1}\tilde{\boldsymbol{\delta}} t + H^{-1}\sum_{i=0}^{t} \boldsymbol{\epsilon}_i] + \mathbf{a}_t \\
&= \mathbf{y}_0 + A\boldsymbol{\tau}_t + \mathbf{a}_t
\end{aligned}$$

and

$$\boldsymbol{\tau}_t = \pi + \boldsymbol{\tau}_{t-1} + \mathbf{v}_t$$

where $\mathbf{a}_t = \Psi^*(B)\boldsymbol{\epsilon}_t$, $\pi = S_m H^{-1}\tilde{\boldsymbol{\delta}}$, $\boldsymbol{\tau}_t = S_m[H^{-1}\tilde{\boldsymbol{\delta}} t + H^{-1}\sum_{i=0}^{t} \boldsymbol{\epsilon}_i]$, and $\mathbf{v}_t = S_m H^{-1}\boldsymbol{\epsilon}_t$.

Stock and Watson showed that the common trends representation expresses $\mathbf{y}_t$ as a linear combination of m random walks ($\boldsymbol{\tau}_t$) with drift π plus $I(0)$ components ($\mathbf{a}_t$).

Test for the Common Trends

Stock and Watson (1988) proposed statistics for common trends testing. The null hypothesis is that k-dimensional time series $\mathbf{y}_t$ has $m \leq k$ common stochastic trends, and the alternative is that it has $s < m$ common trends. The test procedure of m vs s common stochastic trends is performed based on the first-order serial correlation matrix of $\mathbf{y}_t$. Let $\boldsymbol{\beta}_\perp$ be a $k \times m$ matrix orthogonal to the cointegrating matrix such that $\boldsymbol{\beta}_\perp^{'}\boldsymbol{\beta} = 0$, and $\boldsymbol{\beta}_\perp\boldsymbol{\beta}_\perp^{'} = I_m$. Let $\mathbf{z}_t = \boldsymbol{\beta}'\mathbf{y}_t$ and $\mathbf{w}_t = \boldsymbol{\beta}_\perp^{'}\mathbf{y}_t$. Then

$$\mathbf{w}_t = \boldsymbol{\beta}'_\perp\mathbf{y}_0 + \boldsymbol{\beta}'_\perp\boldsymbol{\delta}t + \boldsymbol{\beta}'_\perp\Psi(1)\sum_{i=0}^{t}\boldsymbol{\epsilon}_i + \boldsymbol{\beta}'_\perp\Psi^*(B)\boldsymbol{\epsilon}_t$$

Combining the expression of $\mathbf{z}_t$ and $\mathbf{w}_t$,

$$\begin{aligned}\begin{bmatrix}\mathbf{z}_t\\ \mathbf{w}_t\end{bmatrix} &= \begin{bmatrix}\boldsymbol{\beta}'\mathbf{y}_0\\ \boldsymbol{\beta}_\perp^{'}\mathbf{y}_0\end{bmatrix} + \begin{bmatrix}0\\ \boldsymbol{\beta}_\perp^{'}\boldsymbol{\delta}\end{bmatrix}t + \begin{bmatrix}0\\ \boldsymbol{\beta}_\perp^{'}\Psi(1)\end{bmatrix}\sum_{i=1}^{t}\boldsymbol{\epsilon}_i\\ &+ \begin{bmatrix}\boldsymbol{\beta}'\Psi^*(B)\\ \boldsymbol{\beta}'_\perp\Psi^*(B)\end{bmatrix}\boldsymbol{\epsilon}_t\end{aligned}$$

The Stock-Watson common trends test is performed based on the component $\mathbf{w}_t$ by testing whether $\boldsymbol{\beta}_\perp^{'}\Psi(1)$ has rank m against rank s.

The following statements test Stock-Watson common trends:

```
proc varmax data=simul2;
   model y1 y2 / p=2 cointtest=(sw);
run;
```

```
                              The VARMAX Procedure

                       Testing for Stock-Watson's Common
                        Trends Using Differencing Filter

                                                             5%
      H0:          H1:                                    Critical
    Rank=m       Rank=s      Eigenvalue      Filter        Value       Lag

         1            0        1.000906        0.09       -14.10         2
         2            0        0.996763       -0.32        -8.80
                      1        0.648908      -35.11       -23.00
```

Figure 30.46. Common Trends Test (COINTTEST=(SW) Option)

In Figure 30.46, the first column is the null hypothesis that $\mathbf{y}_t$ has $m \leq k$ common trends; the second column, the alternative hypothesis that $\mathbf{y}_t$ has $s < m$ common trends; the fourth column, the test statistics using AR(2) filtering the data. The test statistics for testing for 2 versus 1 common trends are more negative (-35.1) than the critical value (-23.0). The test rejects the null hypothesis, which means that the series has a single common trend.

Vector Error Correction Modeling

This section discusses the implication of cointegration for the autoregressive representation. Assume that the cointegrated series can be represented by a vector error correction model according to the Granger representation theorem (Engle and Granger 1987). Consider the vector autoregressive process with Gaussian errors defined by

$$\mathbf{y}_t = \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

or

$$\Phi(B)\mathbf{y}_t = \boldsymbol{\epsilon}_t$$

where the initial values, $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$, are fixed and $\boldsymbol{\epsilon}_t \sim N(0, \Sigma)$. Since the AR operator $\Phi(B)$ can be re-expressed as $\Phi(B) = \Phi^*(B)(1-B) + \Phi(1)B$ where $\Phi^*(B) = I_k - \sum_{i=1}^{p-1} \Phi_i^* B^i$ with $\Phi_i^* = -\sum_{j=i+1}^{p} \Phi_j$, the vector error correction model is

$$\Phi^*(B)(1-B)\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

or

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta\mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{\alpha}\boldsymbol{\beta}' = -\Phi(1) = -I_k + \Phi_1 + \Phi_2 + \cdots + \Phi_p$.

One motivation for the VECM(p) form is to consider the relation $\boldsymbol{\beta}'\mathbf{y}_t = \mathbf{c}$ as defining the underlying economic relations and assume that the agents react to the disequilibrium error $\boldsymbol{\beta}'\mathbf{y}_t - \mathbf{c}$ through the adjustment coefficient $\boldsymbol{\alpha}$ to restore equilibrium; that is, they satisfy the economic relations. The cointegrating vector, $\boldsymbol{\beta}$ is sometimes called the *long-run parameters*.

You can consider a vector error correction model with a deterministic term. The deterministic term D_t can contain a constant, a linear trend, seasonal dummy variables, or nonstochastic regressors.

$$\Delta\mathbf{y}_t = \Pi\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta\mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

where $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$.

The alternative vector error correction representation considers the error correction term at lag $t-p$ and is written as

$$\Delta\mathbf{y}_t = \sum_{i=1}^{p-1} \Phi_i^{\sharp} \Delta\mathbf{y}_{t-i} + \Pi^{\sharp}\mathbf{y}_{t-p} + AD_t + \boldsymbol{\epsilon}_t$$

If the matrix Π has a full-rank ($r = k$), all components of $\mathbf{y}_t$ are $I(0)$. On the other hand, $\mathbf{y}_t$ are stationary in difference if $rank(\Pi) = 0$. When the rank of the matrix Π is $r < k$, there are $k - r$ linear combinations that are nonstationary and r stationary cointegrating relations. Note that the linearly independent vector $\mathbf{z}_t = \boldsymbol{\beta}'\mathbf{y}_t$ is stationary and this transformation is not unique unless $r = 1$. There does not exist a unique cointegrating matrix $\boldsymbol{\beta}$ since the coefficient matrix Π can also be decomposed as

$$\Pi = \boldsymbol{\alpha} M M^{-1} \boldsymbol{\beta}' = \boldsymbol{\alpha}^* \boldsymbol{\beta}^{*'}$$

where M is an $r \times r$ nonsingular matrix.

Test for the Cointegration

The cointegration rank test determines the linearly independent columns of Π. Johansen (1988, 1995a) and Johansen and Juselius (1990) proposed the cointegration rank test using the reduced rank regression.

Different Specifications of Deterministic Trends

When you construct the VECM(p) form from the VAR(p) model, the deterministic terms in the VECM(p) form can differ from those in the VAR(p) model. When there are deterministic cointegrated relationships among variables, deterministic terms in the VAR(p) model will not be present in the VECM(p) form. On the other hand, if there are stochastic cointegrated relationships, deterministic terms appear in the VECM(p) form via the error correction term or as an independent term in the VECM(p) form.

- **Case 1:** There is no separate drift in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

- **Case 2:** There is no separate drift in the VECM(p) form, but a constant enters only via the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \beta_0)(\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

- **Case 3:** There is a separate drift and no separate linear trend in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

- **Case 4:** There is a separate drift and no separate linear trend in the VECM(p) form, but a linear trend enters only via the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \beta_1)(\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

- **Case 5:** There is a separate linear trend in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 t + \boldsymbol{\epsilon}_t$$

First, focus on cases 1, 3, and 5 to test the null hypothesis that there are at most r cointegrating vectors. Let

$$\begin{aligned} Z_{0t} &= \Delta \mathbf{y}_t \\ Z_{1t} &= \mathbf{y}_{t-1} \\ Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \ldots, \Delta \mathbf{y}'_{t-p+1}, D_t]' \\ Z_0 &= [Z_{01}, \ldots, Z_{0T}]' \\ Z_1 &= [Z_{11}, \ldots, Z_{1T}]' \\ Z_2 &= [Z_{21}, \ldots, Z_{2T}]' \end{aligned}$$

where D_t can be empty for Case 1; 1 for Case 3; $(1, t)$ for Case 5.

In case 2, Z_{1t} and Z_{2t} are defined as

$$\begin{aligned} Z_{1t} &= [\mathbf{y}'_{t-1}, 1]' \\ Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \ldots, \Delta \mathbf{y}'_{t-p+1}]' \end{aligned}$$

In case 4, Z_{1t} and Z_{2t} are defined as

$$\begin{aligned} Z_{1t} &= [\mathbf{y}'_{t-1}, t]' \\ Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \ldots, \Delta \mathbf{y}'_{t-p+1}, 1]' \end{aligned}$$

Let Ψ be the matrix of parameters consisting of $\Phi_1^*, \ldots, \Phi_{p-1}^*$ and A, where parameters A corresponds to regressors D_t. Then the VECM(p) form is rewritten in these variables as

$$Z_{0t} = \boldsymbol{\alpha}\boldsymbol{\beta}' Z_{1t} + \Psi Z_{2t} + \boldsymbol{\epsilon}_t$$

The log-likelihood function is given by

$$\begin{aligned}\ell &= -\frac{kT}{2}\log 2\pi - \frac{T}{2}\log|\Sigma| \\ &\quad -\frac{1}{2}\sum_{t=1}^{T}(Z_{0t}-\boldsymbol{\alpha\beta}'Z_{1t}-\Psi Z_{2t})'\Sigma^{-1}(Z_{0t}-\boldsymbol{\alpha\beta}'Z_{1t}-\Psi Z_{2t})\end{aligned}$$

The residuals, R_{0t} and R_{1t}, are obtained by regressing Z_{0t} and Z_{1t} on Z_{2t}, respectively. The regression equation in residuals is

$$R_{0t} = \boldsymbol{\alpha\beta}'R_{1t} + \hat{\boldsymbol{\epsilon}}_t$$

The crossproducts matrices are computed

$$S_{ij} = \frac{1}{T}\sum_{t=1}^{T} R_{it}R'_{jt}, \;\; i,j = 0,1$$

Then the maximum likelihood estimator for $\boldsymbol{\beta}$ is obtained from the eigenvectors corresponding to the r largest eigenvalues of the following equation:

$$|\lambda S_{11} - S_{10}S_{00}^{-1}S_{01}| = 0$$

The eigenvalues of the preceding equation are squared canonical correlations between R_{0t} and R_{1t}, and the eigenvectors corresponding to the r largest eigenvalues are the r linear combinations of $\mathbf{y}_{t-1}$, which have the largest squared partial correlations with the stationary process $\Delta\mathbf{y}_t$ after correcting for lags and deterministic terms. Such an analysis calls for a reduced rank regression of $\Delta\mathbf{y}_t$ on $\mathbf{y}_{t-1}$ corrected for $(\Delta\mathbf{y}_{t-1},\ldots,\Delta\mathbf{y}_{t-p+1},D_t)$, as discussed by Anderson (1951). Johansen (1988) suggested two test statistics to test the null hypothesis that there are at most r cointegrating vectors

$$\mathrm{H}_0 : \lambda_i = 0 \;\text{ for }\; i = r+1,\ldots,k$$

Trace Test

$$\lambda_{trace} = -T\sum_{i=r+1}^{k}\log(1-\lambda_i)$$

The asymptotic distribution of this statistic is given by

$$tr\left\{\int_0^1 (dW)\tilde{W}'\left(\int_0^1 \tilde{W}\tilde{W}'dr\right)^{-1}\int_0^1 \tilde{W}(dW)'\right\}$$

where $tr(A)$ is the trace of a matrix A, W is the $k-r$ dimensional Brownian motion, and $\tilde{W}$ is the Brownian motion itself, or the demeaned or detrended Brownian motion according to the different specifications of deterministic trends in the vector error correction model.

Maximum Eigenvalue Test

$$\lambda_{max} = -T \log(1 - \lambda_{r+1})$$

The asymptotic distribution of this statistic is given by

$$max\{\int_0^1 (dW)\tilde{W}'(\int_0^1 \tilde{W}\tilde{W}'dr)^{-1}\int_0^1 \tilde{W}(dW)'\}$$

where $max(A)$ is the maximum eigenvalue of a matrix A. Osterwald-Lenum (1992) provided the detailed tables of critical values of these statistics.

In case 2, consider that $Z_{1t} = (\mathbf{y}_t', 1)'$ and $Z_{2t} = (\Delta\mathbf{y}_{t-1}', \ldots, \Delta\mathbf{y}_{t-p+1}')'$. In case 4, $Z_{1t} = (\mathbf{y}_t', t)'$ and $Z_{2t} = (\Delta\mathbf{y}_{t-1}', \ldots, \Delta\mathbf{y}_{t-p+1}', 1)'$.

The following statements use the JOHANSEN option to compute the Johansen cointegration rank test of integrated order 1:

```
proc varmax data=simul2;
   model y1 y2 / p=2 cointtest=(johansen=(normalize=y1));
run;
```

```
                              The VARMAX Procedure

                       Cointegration Rank Test Using Trace

                                                      5%
  H0:      H1:                                    Critical    Drift       Drift in
 Rank=r   Rank>r   Eigenvalue         Trace          Value    in ECM      Process

      0        0       0.4644       61.7522          15.34    Constant    Linear
      1        1       0.0056        0.5552           3.84


               Cointegration Rank Test Using Trace Under Restriction

                                                      5%
  H0:      H1:                                    Critical    Drift       Drift in
 Rank=r   Rank>r   Eigenvalue         Trace          Value    in ECM      Process

      0        0       0.5209       76.3788          19.99    Constant    Constant
      1        1       0.0426        4.2680           9.13
```

Figure 30.47. Cointegration Rank Test (COINTTEST=(JOHANSEN=) Option)

Suppose that the model has an intercept term. The first table in Figure 30.47 shows the trace statistics based on case 3; the second table, case 2. The output indicates that the series are cointegrated with rank 1.

```
                              The VARMAX Procedure

                         Hypothesis of the Restriction

                                      Drift         Drift in
                       Hypothesis     in ECM        Process

                       H0             Constant      Constant
                       H1             Constant      Linear


                        Hypothesis Test of the Restriction

                                   Restricted
          Rank     Eigenvalue      Eigenvalue         DF     Chi-Square     Pr > ChiSq

             0         0.4644          0.5209          2          14.63         0.0007
             1         0.0056          0.0426          1           3.71         0.0540
```

Figure 30.48. Cointegration Rank Test Continued

Figure 30.48 shows which result, either case 2 (the hypothesis H_0) or case 3 (the hypothesis H_1), is appropriate. Since the cointegration rank is chosen to be 1 by the result in Figure 30.47, look at the last row corresponding to rank=1. Since the p-value is 0.054, the case 2 cannot be rejected at the significance level 5%, but it can be rejected at the significance level 10%. For modeling of two cases, see Figure 30.51 and Figure 30.52.

```
                              The VARMAX Procedure

                        Long-Run Parameter Beta Estimates

                     Variable                1               2

                     y1                1.00000         1.00000
                     y2               -2.04869        -0.02854


                      Adjustment Coefficient Alpha Estimates

                     Variable                1               2

                     y1               -0.46421        -0.00502
                     y2                0.17535        -0.01275
```

Figure 30.49. Cointegration Rank Test Continued

Figure 30.49 shows the estimates of long-run parameter and adjustment coefficients based on case 3. Considering that the cointegration rank is 1, the long-run relationship of the series is

$$y_{1t} = 2.049\; y_{2t}$$

```
                              The VARMAX Procedure

                          Long-Run Coefficient Beta Based
                              on the Restricted Trend

                    Variable                 1                2

                    y1                 1.00000          1.00000
                    y2                -2.04366         -2.75773
                    1                  6.75919        101.37051


                            Adjustment Coefficient Alpha
                           Based on the Restricted Trend

                    Variable                 1                2

                    y1                -0.48015          0.01091
                    y2                 0.12538          0.03722
```

Figure 30.50. Cointegration Rank Test Continued

Figure 30.50 shows the estimates of long-run parameter and adjustment coefficients based on case 2. Considering that the cointegration rank is 1, the long-run relationship of the series is

$$y_{1t} = 2.044\, y_{2t} - 6.760$$

Estimation of Vector Error Correction Model

The preceding log-likelihood function is maximized for

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= S_{11}^{-1/2}[v_1, \ldots, v_r] \\
\hat{\boldsymbol{\alpha}} &= S_{01}\hat{\boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}' S_{11}\hat{\boldsymbol{\beta}})^{-1} \\
\hat{\Pi} &= \hat{\boldsymbol{\alpha}}\hat{\boldsymbol{\beta}}' \\
\hat{\Psi} &= (Z_2' Z_2)^{-1} Z_2'(Z_0 - Z_1\hat{\Pi}') \\
\hat{\Sigma} &= (Z_0 - Z_2\hat{\Psi}' - Z_1\hat{\Pi}')'(Z_0 - Z_2\hat{\Psi}' - Z_1\hat{\Pi}')/T
\end{aligned}$$

The estimators of the orthogonal complements of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are

$$\hat{\boldsymbol{\beta}}_{\perp} = S_{11}[v_{r+1}, \ldots, v_k]$$

and

$$\hat{\boldsymbol{\alpha}}_{\perp} = S_{00}^{-1} S_{01}[v_{r+1}, \ldots, v_k]$$

The ML estimators have the following asymptotic properties:

$$\sqrt{T}\mathrm{vec}([\hat{\Pi}, \hat{\Psi}] - [\Pi, \Psi]) \xrightarrow{d} N(0, \Sigma_{co})$$

where

$$\Sigma_{co} = \Sigma \otimes \left(\left[\begin{array}{cc} \boldsymbol{\beta} & 0 \\ 0 & I_k \end{array} \right] \Omega^{-1} \left[\begin{array}{cc} \boldsymbol{\beta}' & 0 \\ 0 & I_k \end{array} \right] \right)$$

and

$$\Omega = \mathrm{plim} \frac{1}{T} \left[\begin{array}{cc} \boldsymbol{\beta}' Z_1' Z_1 \boldsymbol{\beta} & \boldsymbol{\beta}' Z_1' Z_2 \\ Z_2' Z_1 \boldsymbol{\beta} & Z_2' Z_2 \end{array} \right]$$

The following statements are the examples of fitting the different types of the vector error correction models mentioned in the previous section.

For fitting case 1,

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1) noint;
```

For fitting case 2,

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1 ectrend);
```

For fitting case 3,

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
```

For fitting case 4,

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1 ectrend)
              trend=linear;
```

For fitting case 5,

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1) trend=linear;
```

From Figure 30.48 on page 1791 using the COINTTEST=(JOHANSEN) option, you can fit the model using either case 2 or case 3. Here both models are fitted to show the difference in output display. Figure 30.51 is for case 2 and Figure 30.52 is for case 3.

For case 2,

```
proc varmax data=simul2;
   model y1 y2 / p=2 ecm=(rank=1 normalize=y1 ectrend);
run;
```

```
                              The VARMAX Procedure

                       Parameter Alpha * Beta' Estimates

             Variable              y1              y2                1

             y1              -0.48015         0.98126         -3.24543
             y2               0.12538        -0.25624          0.84748


                       AR Coefficients of Differenced Lag

               DIF Lag    Variable              y1              y2

                     1    y1              -0.72759        -0.77463
                          y2               0.38982        -0.55173


                           Model Parameter Estimates

                                           Standard
Equation Parameter      Estimate              Error  t Value Pr > |t| Variable

D_y1     CONST1         -3.24543      0.33022                 1, EC
         AR1_1_1        -0.48015      0.04886                 y1(t-1)
         AR1_1_2         0.98126      0.09984                 y2(t-1)
         AR2_1_1        -0.72759      0.04623  -15.74   0.0001 D_y1(t-1)
         AR2_1_2        -0.77463      0.04978  -15.56   0.0001 D_y2(t-1)
D_y2     CONST2          0.84748      0.35394                 1, EC
         AR1_2_1         0.12538      0.05236                 y1(t-1)
         AR1_2_2        -0.25624      0.10702                 y2(t-1)
         AR2_2_1         0.38982      0.04955    7.87   0.0001 D_y1(t-1)
         AR2_2_2        -0.55173      0.05336  -10.34   0.0001 D_y2(t-1)
```

Figure 30.51. Parameter Estimation with the ECTREND Option

Figure 30.51 can be reported as follows:

$$\Delta \mathbf{y}_t = \left[\begin{array}{rrr} -0.48015 & 0.98126 & -3.24543 \\ 0.12538 & -0.25624 & 0.84748 \end{array}\right] \left[\begin{array}{c} y_{1,t-1} \\ y_{2,t-1} \\ 1 \end{array}\right] + \left[\begin{array}{rr} -0.72759 & -0.77463 \\ 0.38982 & -0.55173 \end{array}\right] \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The keyword "EC" in the model parameter estimation table means that the ECTREND option is used for fitting the model.

For fitting case 3,

```
proc varmax data=simul2;
   model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
run;
```

```
                          The VARMAX Procedure

                  Parameter Alpha * Beta' Estimates

               Variable              y1              y2

               y1              -0.46421         0.95103
               y2               0.17535        -0.35923


                  AR Coefficients of Differenced Lag

         DIF Lag    Variable              y1              y2

               1    y1              -0.74052        -0.76305
                    y2               0.34820        -0.51194


                      Model Parameter Estimates

                                           Standard
Equation Parameter      Estimate           Error t Value Pr > |t| Variable

D_y1     CONST1         -2.60825        1.32398   -1.97   0.0518 1
         AR1_1_1        -0.46421        0.05474                  y1(t-1)
         AR1_1_2         0.95103        0.11215                  y2(t-1)
         AR2_1_1        -0.74052        0.05060  -14.63   0.0001 D_y1(t-1)
         AR2_1_2        -0.76305        0.05352  -14.26   0.0001 D_y2(t-1)
D_y2     CONST2          3.43005        1.39587    2.46   0.0159 1
         AR1_2_1         0.17535        0.05771                  y1(t-1)
         AR1_2_2        -0.35923        0.11824                  y2(t-1)
         AR2_2_1         0.34820        0.05335    6.53   0.0001 D_y1(t-1)
         AR2_2_2        -0.51194        0.05643   -9.07   0.0001 D_y2(t-1)
```

Figure 30.52. Parameter Estimation without the ECTREND Option

Figure 30.52 can be reported as follows:

$$\begin{aligned}\Delta \mathbf{y}_t &= \left[\begin{array}{rr} -0.46421 & 0.95103 \\ 0.17535 & -0.35293 \end{array}\right] \mathbf{y}_{t-1} + \left[\begin{array}{rr} -0.74052 & -0.76305 \\ 0.34820 & -0.51194 \end{array}\right] \Delta \mathbf{y}_{t-1} \\ &+ \left[\begin{array}{r} -2.60825 \\ 3.43005 \end{array}\right] + \boldsymbol{\epsilon}_t\end{aligned}$$

Test for the Linear Restriction of β

Consider the example with the variables m_t, log real money, y_t, log real income, i_t^d, deposit interest rate, and i_t^b, bond interest rate. It seems a natural hypothesis that in the long-run relation, money and income have equal coefficients with opposite signs. This can be formulated as the hypothesis that the cointegrated relation contains only m_t and y_t through $m_t - y_t$. For the analysis, you can express these restrictions in the parameterization of H such that $\boldsymbol{\beta} = H\psi$, where H is a known $k \times s$ matrix and ψ is the $s \times r (r \leq s < k)$ parameter matrix to be estimated. For this example, H is given by

$$H = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Restriction H_0: $\boldsymbol{\beta} = H\phi$

When the linear restriction $\boldsymbol{\beta} = H\phi$ is given, it implies that the same restrictions are imposed on all cointegrating vectors. You obtain the maximum likelihood estimator of $\boldsymbol{\beta}$ by reduced rank regression of $\Delta\mathbf{y}_t$ on $H\mathbf{y}_{t-1}$ corrected for $(\Delta\mathbf{y}_{t-1}, \ldots, \Delta\mathbf{y}_{t-p+1}, D_t)$, solving the following equation

$$|\rho H' S_{11} H - H' S_{10} S_{00}^{-1} S_{01} H| = 0$$

for the eigenvalues $1 > \rho_1 > \cdots > \rho_s > 0$ and eigenvectors $(v_1, \ldots, v_s)$, S_{ij} are given in the preceding section. Then choose $\hat{\phi} = (v_1, \ldots, v_r)$ corresponding to the r largest eigenvalues, and the $\hat{\boldsymbol{\beta}}$ is $H\hat{\phi}$.

The test statistic for H_0: $\boldsymbol{\beta} = H\phi$ is given by

$$T \sum_{i=1}^{r} \log\{(1-\rho_i)/(1-\lambda_i)\} \xrightarrow{d} \chi^2_{r(k-s)}$$

If the series has no deterministic trend, the constant term should be restricted by $\boldsymbol{\alpha}'_{\perp}\boldsymbol{\delta}_0 = 0$ like Case 2. Then H is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following statements test that $\beta_1 + 2\beta 2 = 0$:

```
proc varmax data=simul2;
   model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
   cointeg rank=1 h=(1,-2);
run;
```

```
                            The VARMAX Procedure

                            Long-Run Coefficient
                            Beta with Respect to
                             Hypothesis on Beta

                          Variable              1

                          y1              1.00000
                          y2             -2.00000


                           Adjustment Coefficient
                           Alpha with Respect to
                             Hypothesis on Beta

                          Variable              1

                          y1             -0.47404
                          y2              0.17534


             Test for Restricted Long-Run Coefficient Beta

                              Restricted
     Index     Eigenvalue     Eigenvalue        DF     Chi-Square     Pr > ChiSq

         1         0.4644         0.4616         1           0.51         0.4738
```

Figure 30.53. Testing of Linear Restriction $\boldsymbol{\beta}$ (H= Option)

Figure 30.53 shows the results of testing $H_0\colon \beta_1 + 2\beta 2 = 0$. The input H matrix is $H = (1, -2)'$. The adjustment coefficient is reestimated under the restriction, and the test indicates that you cannot reject the null hypothesis.

Test for the Weak Exogeneity and Restrictions of α

Consider a vector error correction model:

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

Divide the process $\mathbf{y}_t$ into $(\mathbf{y}_{1t}', \mathbf{y}_{2t}')'$ with dimension k_1 and k_2 and the Σ into

$$\Sigma = \left[\begin{array}{cc} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{array} \right]$$

Similarly, the parameters can be decomposed as follows:

$$\boldsymbol{\alpha} = \left[\begin{array}{c} \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \end{array} \right] \quad \Phi_i^* = \left[\begin{array}{c} \Phi_{1i}^* \\ \Phi_{2i}^* \end{array} \right] \quad A = \left[\begin{array}{c} A_1 \\ A_2 \end{array} \right]$$

Then the VECM(p) form can be rewritten using the decomposed parameters and processes:

$$\begin{bmatrix} \Delta\mathbf{y}_{1t} \\ \Delta\mathbf{y}_{2t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \end{bmatrix} \boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \begin{bmatrix} \Phi_{1i}^* \\ \Phi_{2i}^* \end{bmatrix} \Delta\mathbf{y}_{t-i} + \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} D_t + \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

The conditional model for $\mathbf{y}_{1t}$ given $\mathbf{y}_{2t}$ is

$$\begin{aligned} \Delta\mathbf{y}_{1t} &= \omega\Delta\mathbf{y}_{2t} + (\alpha_1 - \omega\alpha_2)\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1}(\Phi_{1i}^* - \omega\Phi_{2i}^*)\Delta\mathbf{y}_{t-i} \\ &\quad +(A_1 - \omega A_2)D_t + \boldsymbol{\epsilon}_{1t} - \omega\boldsymbol{\epsilon}_{2t} \end{aligned}$$

and the marginal model of $\mathbf{y}_{2t}$,

$$\Delta\mathbf{y}_{2t} = \alpha_2\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_{2i}^*\Delta\mathbf{y}_{t-i} + A_2 D_t + \boldsymbol{\epsilon}_{2t}$$

where $\omega = \Sigma_{12}\Sigma_{22}^{-1}$.

The test of weak exogeneity of $\mathbf{y}_{2t}$ for the parameters $(\alpha_1, \boldsymbol{\beta})$ determines whether $\alpha_2 = 0$. Weak exogeneity means that there is no information about $\boldsymbol{\beta}$ in the marginal model or that the variables $\mathbf{y}_{2t}$ do not react to a disequilibrium.

Restriction H_0: $\boldsymbol{\alpha} = J\psi$

Consider the null hypothesis H_0: $\boldsymbol{\alpha} = J\psi$, where J is a $k \times m$ matrix with $r \leq m < k$.

From the previous residual regression equation

$$R_{0t} = \boldsymbol{\alpha}\boldsymbol{\beta}'R_{1t} + \hat{\boldsymbol{\epsilon}}_t = J\psi\boldsymbol{\beta}'R_{1t} + \hat{\boldsymbol{\epsilon}}_t$$

you can obtain

$$\begin{aligned} \bar{J}'R_{0t} &= \psi\boldsymbol{\beta}'R_{1t} + \bar{J}'\hat{\boldsymbol{\epsilon}}_t \\ J_\perp' R_{0t} &= J_\perp'\hat{\boldsymbol{\epsilon}}_t \end{aligned}$$

where $\bar{J} = J(J'J)^{-1}$ and $J_\perp$ is orthogonal to J such that $J_\perp' J = 0$.

Define

$$\Sigma_{JJ_\perp} = \bar{J}'\Sigma J_\perp \text{ and } \Sigma_{J_\perp J_\perp} = J_\perp'\Sigma J_\perp$$

and let $\omega = \Sigma_{JJ_\perp}\Sigma_{J_\perp J_\perp}^{-1}$. Then $\bar{J}'R_{0t}$ can be written

$$\bar{J}'R_{0t} = \psi\boldsymbol{\beta}'R_{1t} + \omega J_\perp' R_{0t} + \bar{J}'\hat{\boldsymbol{\epsilon}}_t - \omega J_\perp'\hat{\boldsymbol{\epsilon}}_t$$

Using the marginal distribution of $J_\perp' R_{0t}$ and the conditional distribution of $\bar{J}' R_{0t}$, the new residuals are computed as

$$\begin{aligned}\tilde{R}_{Jt} &= \bar{J}' R_{0t} - S_{JJ_\perp} S_{J_\perp J_\perp}^{-1} J_\perp' R_{0t} \\ \tilde{R}_{1t} &= R_{1t} - S_{1J_\perp} S_{J_\perp J_\perp}^{-1} J_\perp' R_{0t}\end{aligned}$$

where

$$S_{JJ_\perp} = \bar{J}' S_{00} J_\perp, \;\; S_{J_\perp J_\perp} = J_\perp' S_{00} J_\perp, \;\; \text{and} \;\; S_{J_\perp 1} = J_\perp' S_{01}$$

In terms of $\tilde{R}_{Jt}$ and $\tilde{R}_{1t}$, the MLE of $\boldsymbol{\beta}$ is computed by using the reduced rank regression. Let

$$S_{ij.J_\perp} = \frac{1}{T}\sum_{t=1}^{T} \tilde{R}_{it}\tilde{R}_{jt}', \;\; \text{for} \;\; i,j = 1, J$$

Under the null hypothesis $H_0 \colon \boldsymbol{\alpha} = J\psi$, the MLE $\tilde{\boldsymbol{\beta}}$ is computed by solving the equation

$$|\rho S_{11.J_\perp} - S_{1J.J_\perp} S_{JJ.J_\perp}^{-1} S_{J1.J_\perp}| = 0$$

Then $\tilde{\boldsymbol{\beta}} = (v_1, \ldots, v_r)$, where the eigenvectors correspond to the r largest eigenvalues. The likelihood ratio test for $H_0 \colon \boldsymbol{\alpha} = J\psi$ is

$$T\sum_{i=1}^{r} \log\{(1-\rho_i)/(1-\lambda_i)\} \xrightarrow{d} \chi^2_{r(k-m)}$$

The test of weak exogeneity of $\mathbf{y}_{2t}$ is the special case of the test $\boldsymbol{\alpha} = J\psi$, considering $J = (I_{k_1}, 0)'$. Consider the previous example with four variables (m_t, y_t, i_t^b, i_t^d). If $r = 1$, you formulate the weak exogeneity of (y_t, i_t^b, i_t^d) for m_t as $J = [1, 0, 0, 0]'$ and the weak exogeneity of i_t^d for (m_t, y_t, i_t^b) as $J = [I_3, 0]'$.

The following statements test the weak exogeneity of other variables:

```
proc varmax data=simul2;
   model y1 y2 / p=2 ecm=(rank=1 normalize=y1);
   cointeg rank=1 exogeneity;
run;
```

```
                      The VARMAX Procedure

            Testing Weak Exogeneity of Each Variables

          Variable        DF    Chi-Square    Pr > ChiSq

          y1               1         53.46        <.0001
          y2               1          8.76        0.0031
```

Figure 30.54. Testing of Weak Exogeneity (EXOGENEITY Option)

Figure 30.54 shows that each variable is not the weak exogeneity of other variable.

Forecasting of the VECM

Consider the cointegrated moving-average representation of the differenced process of $\mathbf{y}_t$

$$\Delta\mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t$$

Assume that $\mathbf{y}_0 = 0$. The linear process $\mathbf{y}_t$ can be written

$$\mathbf{y}_t = \boldsymbol{\delta}t + \sum_{i=1}^{t}\sum_{j=0}^{t-i}\Psi_j\boldsymbol{\epsilon}_i$$

Therefore, for any $l > 0$,

$$\mathbf{y}_{t+l} = \boldsymbol{\delta}(t+l) + \sum_{i=1}^{t}\sum_{j=0}^{t+l-i}\Psi_j\boldsymbol{\epsilon}_i + \sum_{i=1}^{l}\sum_{j=0}^{l-i}\Psi_j\boldsymbol{\epsilon}_{t+i}$$

The l-step-ahead forecast is derived from the preceding equation:

$$\mathbf{y}_{t+l|t} = \boldsymbol{\delta}(t+l) + \sum_{i=1}^{t}\sum_{j=0}^{t+l-i}\Psi_j\boldsymbol{\epsilon}_i$$

Note that

$$\lim_{l\to\infty}\boldsymbol{\beta}'\mathbf{y}_{t+l|t} = 0$$

since $\lim_{l\to\infty}\sum_{j=0}^{t+l-i}\Psi_j = \Psi(1)$ and $\boldsymbol{\beta}'\Psi(1) = 0$. The long-run forecast of the cointegrated system shows that the cointegrated relationship holds, though there might exist some deviations from the equilibrium status in the short-run. The covariance matrix of the predict error $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t}$ is

$$\Sigma(l) = \sum_{i=1}^{l}[(\sum_{j=0}^{l-i}\Psi_j)\Sigma(\sum_{j=0}^{l-i}\Psi_j')]$$

When the linear process is represented as a VECM(p) model, you can obtain

$$\Delta\mathbf{y}_t = \Pi\mathbf{y}_{t-1} + \sum_{j=1}^{p-1}\Phi_j^*\Delta\mathbf{y}_{t-j} + \boldsymbol{\delta} + \boldsymbol{\epsilon}_t$$

The transition equation is defined as

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + \mathbf{e}_t$$

where $\mathbf{z}_t = (\mathbf{y}'_{t-1}, \Delta\mathbf{y}'_t, \Delta\mathbf{y}'_{t-1}, \cdots, \Delta\mathbf{y}'_{t-p+2})'$ is a state vector and the transition matrix is

$$F = \begin{bmatrix} I_k & I_k & 0 & \cdots & 0 \\ \Pi & (\Pi + \Phi_1^*) & \Phi_2^* & \cdots & \Phi_{p-1}^* \\ 0 & I_k & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

where 0 is a $k \times k$ zero matrix. The observation equation can be written

$$\mathbf{y}_t = \boldsymbol{\delta} t + H\mathbf{z}_t$$

where $H = [I_k, I_k, 0, \ldots, 0]$.

The l-step-ahead forecast is computed as

$$\mathbf{y}_{t+l|t} = \boldsymbol{\delta}(t+l) + HF^l\mathbf{z}_t$$

$\mathrm{I}(2)$ Model

The VAR(p) model can be written as the error correction form.

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^*\Delta\mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

Let $\Phi^* = I_k - \sum_{i=1}^{p-1}\Phi_i^*$. If $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have full-rank r, and if

$$rank(\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp) = k - r$$

then $\mathbf{y}_t$ is an $I(1)$ process. If the condition $rank(\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp) = k - r$ fails and $\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp$ has reduced-rank $\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$ where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r)\times s$ matrices with $s \le k-r$, $\boldsymbol{\alpha}_\perp$ and $\boldsymbol{\beta}_\perp$ are defined as $k \times (k-r)$ matrices of full rank such that $\boldsymbol{\alpha}'\boldsymbol{\alpha}_\perp = 0$ and $\boldsymbol{\beta}'\boldsymbol{\beta}_\perp = 0$. If $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have full-rank s, then the process $\mathbf{y}_t$ is $I(2)$, which has the implication of $I(2)$ model for the moving-average representation.

$$\mathbf{y}_t = B_0 + B_1 t + C_2 \sum_{j=1}^{t}\sum_{i=1}^{j} \boldsymbol{\epsilon}_i + C_1 \sum_{i=1}^{t} \boldsymbol{\epsilon}_i + C_0(B)\boldsymbol{\epsilon}_t$$

The matrices C_1, C_2, and $C_0(B)$ are determined by the cointegration properties of the process, and B_0 and B_1 are determined by the initial values. For details, see Johansen (1995a).

The implication of the $I(2)$ model for the autoregressive representation is given by

$$\Delta^2\mathbf{y}_t = \Pi\mathbf{y}_{t-1} - \Phi^*\Delta\mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i\Delta^2\mathbf{y}_{t-i} + AD_t + \boldsymbol{\epsilon}_t$$

where $\Psi_i = -\sum_{j=i+1}^{p-1}\Phi_i^*$ and $\Phi^* = I_k - \sum_{i=1}^{p-1}\Phi_i^*$.

Test for $\mathrm{I}(2)$

The $I(2)$ cointegrated model is given by the following parameter restrictions:

$$H_{r,s}\colon \Pi = \boldsymbol{\alpha}\boldsymbol{\beta}' \text{ and } \boldsymbol{\alpha}_\perp' \Phi^* \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r)\times s$ matrices with $0 \le s \le k-r$. Let H_r^0 represent the $I(1)$ model where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have full-rank r, let $H_{r,s}^0$ represent the $I(2)$ model where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have full-rank s, and let $H_{r,s}$ represent the $I(2)$ model where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have rank $\le s$. The following table shows the relation between the $I(1)$ models and the $I(2)$ models.

Table 30.1. Relation between the $I(1)$ and $I(2)$ Models

					$I(2)$					$I(1)$	
$r\backslash k-r-s$	k		k-1		$\cdots$		1				
0	H_{00}	$\subset$	H_{01}	$\subset$	$\cdots$	$\subset$	$H_{0,k-1}$	$\subset$	H_{0k}	$=$	H_0^0
1			H_{10}	$\subset$	$\cdots$	$\subset$	$H_{1,k-2}$	$\subset$	$H_{1,k-1}$	$=$	H_1^0
$\vdots$							$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-1$							$H_{k-1,0}$	$\subset$	$H_{k-1,1}$	$=$	H_{k-1}^0

Johansen (1995a) proposed the two-step procedure to analyze the $I(2)$ model. In the first step, the values of $(r, \boldsymbol{\alpha}, \boldsymbol{\beta})$ are estimated using the reduced rank regression analysis, performing the regression analysis $\Delta^2\mathbf{y}_t$, $\Delta\mathbf{y}_{t-1}$, and $\mathbf{y}_{t-1}$ on $\Delta^2\mathbf{y}_{t-1},\ldots,\Delta^2\mathbf{y}_{t-p+2}, D_t$. This gives residuals R_{0t}, R_{1t}, and R_{2t} and residual product moment matrices

$$M_{ij} = \frac{1}{T}\sum_{t=1}^{T} R_{it}R_{jt}' \text{ for } i,j = 0,1,2$$

Perform the reduced rank regression analysis $\Delta^2\mathbf{y}_t$ on $\mathbf{y}_{t-1}$ corrected for $\Delta\mathbf{y}_{t-1}$, $\Delta^2\mathbf{y}_{t-1},\ldots,\Delta^2\mathbf{y}_{t-p+2}, D_t$ and solve the eigenvalue problem of the equation

$$|\lambda M_{22.1} - M_{20.1}M_{00.1}^{-1}M_{02.1}| = 0$$

where $M_{ij.1} = M_{ij} - M_{i1}M_{11}^{-1}M_{1j}$ for $i,j = 0,2$.

In the second step, if $(r, \boldsymbol{\alpha}, \boldsymbol{\beta})$ are known, the values of $(s, \boldsymbol{\xi}, \boldsymbol{\eta})$ are determined using the reduced rank regression analysis, regressing $\hat{\boldsymbol{\alpha}}_\perp'\Delta^2\mathbf{y}_t$ on $\hat{\boldsymbol{\beta}}_\perp'\Delta\mathbf{y}_{t-1}$ corrected for $\Delta^2\mathbf{y}_{t-1},\ldots,\Delta^2\mathbf{y}_{t-p+2}, D_t$ and $\hat{\boldsymbol{\beta}}'\Delta\mathbf{y}_{t-1}$.

The reduced rank regression analysis reduces to the solution of an eigenvalue problem for the equation

$$|\rho M_{\boldsymbol{\beta}_\perp\boldsymbol{\beta}_\perp.\boldsymbol{\beta}} - M_{\boldsymbol{\beta}_\perp\boldsymbol{\alpha}_\perp.\boldsymbol{\beta}}M_{\boldsymbol{\alpha}_\perp\boldsymbol{\alpha}_\perp.\boldsymbol{\beta}}^{-1}M_{\boldsymbol{\alpha}_\perp\boldsymbol{\beta}_\perp.\boldsymbol{\beta}}| = 0$$

where

$$M_{\boldsymbol{\beta}_\perp\boldsymbol{\beta}_\perp.\boldsymbol{\beta}} = \boldsymbol{\beta}_\perp'(M_{11} - M_{11}\boldsymbol{\beta}(\boldsymbol{\beta}'M_{11}\boldsymbol{\beta})^{-1}\boldsymbol{\beta}'M_{11})\boldsymbol{\beta}_\perp$$

$$\begin{aligned} M'_{\boldsymbol{\beta}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} &= M_{\boldsymbol{\alpha}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} = \bar{\boldsymbol{\alpha}}'_\perp (M_{01} - M_{01}\boldsymbol{\beta}(\boldsymbol{\beta}' M_{11}\boldsymbol{\beta})^{-1}\boldsymbol{\beta}' M_{11})\boldsymbol{\beta}_\perp \\ M_{\boldsymbol{\alpha}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} &= \bar{\boldsymbol{\alpha}}'_\perp (M_{00} - M_{01}\boldsymbol{\beta}(\boldsymbol{\beta}' M_{11}\boldsymbol{\beta})^{-1}\boldsymbol{\beta}' M_{10})\bar{\boldsymbol{\alpha}}_\perp \end{aligned}$$

where $\bar{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(\boldsymbol{\alpha}'\boldsymbol{\alpha})^{-1}$.

The solution gives eigenvalues $1 > \rho_1 > \cdots > \rho_s > 0$ and eigenvectors $(v_1, \ldots, v_s)$. Then, the ML estimators are

$$\begin{aligned} \hat{\boldsymbol{\eta}} &= (v_1, \ldots, v_s) \\ \hat{\boldsymbol{\xi}} &= M_{\boldsymbol{\alpha}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} \hat{\eta} \end{aligned}$$

The likelihood ratio test for the reduced rank model $H_{r,s}$ with rank $\leq s$ in the model $H_{r,k-r} = H_r^0$ is given by

$$Q_{r,s} = -T \sum_{i=s+1}^{k-r} \log(1-\rho_i), \;\; s = 0, \ldots, k-r-1$$

The following statements are to test the rank test for the cointegrated order 2:

```
proc varmax data=simul2;
   model y1 y2 / p=2 cointtest=(johansen=(iorder=2));
run;
```

```
                              The VARMAX Procedure

                        Cointegration Rank Test for I(2)

                                                          Trace      5% CV of
      r\k-r-s                 2                1        of I(1)        I(1)

         0           720.40735        308.69199        61.7522         15.34
         1                            211.84512         0.5552          3.84
    5% CV I(2)        15.34000          3.84000
```

Figure 30.55. Cointegrated I(2) Test (IORDER= Option)

The last two columns in Figure 30.55 explain the cointegration rank test with integrated order 1. The results indicate that there is the cointegrated relationship with the cointegration rank 1 with respect to the 0.05 significance level. Now, look at the row in case of $r = 1$. Compare the value to the critical value for the cointegrated order 2. There is no evidence that the series are integrated order 2 with the 0.05 significance level.

Multivariate GARCH Modeling (Experimental)

To study the volatility of time series, GARCH models are widely used since they provide a good approach to conditional variance modeling. In addition, stochastic volatility modeling is important in finance.

Multivariate GARCH

There are three representations of a multivariate GARCH model:

- *BEKK* representation,
- *BEW* representation,
- *diagonal* representation.

Engle and Kroner (1995) proposed a general multivariate GARCH model and called it a *BEKK* representation. Let $\mathcal{F}(t-1)$ be the sigma field generated by the past values of $\boldsymbol{\epsilon}_t$ and H_t be the conditional covariance matrix of the k-dimensional random vector $\boldsymbol{\epsilon}_t$. Let H_t be measurable with respect to $\mathcal{F}(t-1)$, then the multivariate GARCH model can be written as

$$\begin{aligned} \boldsymbol{\epsilon}_t|\mathcal{F}(t-1) &\sim N(0,H_t) \\ H_t &= C_0'C_0 + \sum_{i=1}^{q} A_i'\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}_{t-i}'A_i + \sum_{i=1}^{p} G_i'H_{t-i}G_i \end{aligned}$$

where C_0, A_i and G_i are $k \times k$ parameter matrices with C_0 is an upper triangular matrix.

Consider a bivariate GARCH(1,1) model as follows:

$$\begin{aligned} H_t &= C_0'C_0 + \left[\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array}\right]' \left[\begin{array}{cc} \epsilon_{1,t-1}^2 & \epsilon_{1,t-1}\epsilon_{2,t-1} \\ \epsilon_{2,t-1}\epsilon_{1,t-1} & \epsilon_{2,t-1}^2 \end{array}\right] \left[\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array}\right] \\ &+ \left[\begin{array}{cc} g_{11} & g_{12} \\ g_{21} & g_{22} \end{array}\right]' H_{t-1} \left[\begin{array}{cc} g_{11} & g_{12} \\ g_{21} & g_{22} \end{array}\right] \end{aligned}$$

or, representing the univariate model,

$$\begin{aligned} h_{11,t} &= c_1 + a_{11}^2\epsilon_{1,t-1}^2 + 2a_{11}a_{21}\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{21}^2\epsilon_{2,t-1}^2 \\ &\quad + g_{11}^2h_{11,t-1} + 2g_{11}g_{21}h_{12,t-1} + g_{21}^2h_{22,t-1} \\ h_{12,t} &= c_2 + a_{11}a_{12}\epsilon_{1,t-1}^2 + (a_{21}a_{12}+a_{11}a_{22})\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{21}a_{22}\epsilon_{2,t-1}^2 \\ &\quad + g_{11}g_{12}h_{11,t-1} + (g_{21}g_{12}+g_{11}g_{22})h_{12,t-1} + g_{21}g_{22}h_{22,t-1} \\ h_{22,t} &= c_3 + a_{12}^2\epsilon_{1,t-1}^2 + 2a_{12}a_{22}\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{22}^2\epsilon_{2,t-1}^2 \\ &\quad + g_{12}^2h_{11,t-1} + 2g_{12}g_{22}h_{12,t-1} + g_{22}^2h_{22,t-1} \end{aligned}$$

For the *BEKK* representation, the SAS statement is

```
model y1 y2 / garch=(q=1 p=1 form=bekk);
```

An alternative multivariate GARCH model is suggested by Bollerslev, Engle, and Wooldridge (1988) and is called the *BEW* representation.

$$\text{vech}(H_t) = \mathbf{c}^* + \sum_{i=1}^{q} A_i^* \text{vech}(\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}'_{t-i}) + \sum_{i=1}^{p} G_i^* \text{vech}(H_{t-i})$$

where $\mathbf{c}^*$ is a $k(k+1)/2$-dimensional vector; A_i^* and G_i^* are $k(k+1)/2 \times k(k+1)/2$ matrices.

For the *BEW* representation, the SAS statement is

```
model y1 y2 / garch=(q=1 p=1 form=bew);
```

For empirical implementation, it can be restricted as the diagonal representation and is called the *diagonal* representation. The *diagonal* representation has the A_i^* and G_i^* are $k(k+1)/2 \times k(k+1)/2$ diagonal matrices.

For the *diagonal* representation, the SAS statement is

```
model y1 y2 / garch=(q=1 p=1 form=diag);
```

Applying a multivariate GARCH model to a regression model with the second moments appearing, the model is called a multivariate GARCH-M model. In addition, it can be extended to VAR-GARCH models, GARCH-regression (VX-GARCH) models, and VARX-GARCH models.

$$\begin{aligned}
\Phi(B)\mathbf{y}_t &= \boldsymbol{\delta} + \Theta^*(B)\mathbf{x}_t + \Lambda\mathbf{h}_t + \boldsymbol{\epsilon}_t \\
\boldsymbol{\epsilon}_t|\mathcal{F}(t-1) &\sim N(0, H_t) \\
H_t &= C_0'C_0 + \sum_{i=1}^{q} A_i'\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}'_{t-i}A_i + \sum_{i=1}^{p} G_i'H_{t-i}G_i
\end{aligned}$$

where $\mathbf{y}_t$ is a k-dimensional vector of endogenous variables and $\Phi(B) = I - \Phi_1 B - \cdots - \Phi_{p^0} B^{p^0}$ is a finite order matrix polynomial; $\mathbf{x}_t$ is an m-dimensional vector of exogenous variables and $\Theta^*(B) = \Theta_0^* + \Theta_1^* B + \cdots + \Theta_s^* B^s$ is a finite order matrix polynomial; $\mathbf{h}_t = \text{vech}(H_t)$ and Λ is a $k \times k(k+1)/2$ parameter matrix.

General GARCH models can be also applied to *BEW* and *diagonal* representations.

The following statements are examples of the GARCH= option:

```
model y1 y2 = x1 / p=1 garch=(q=1 p=1 mean form=bekk);
model y1 y2 = x1 / p=1 garch=(q=1 p=1 mean form=bew);
model y1 y2 = x1 / p=1 garch=(q=1 p=1 mean form=diag);
model y1 y2 = x1 / p=1 xlag=1 garch=(q=1);
```

The log-likelihood function of the multivariate GARCH model is written without a constant term

$$\ell = -\frac{1}{2}\sum_{t=1}^{T}[\log|H_t| - \boldsymbol{\epsilon}_t' H_t^{-1}\boldsymbol{\epsilon}_t]$$

The log-likelihood function is maximized by the iterative numerical procedure such as the quasi-Newton optimization. The starting values for the regression parameters are obtained from the least squares estimates; the covariance of $\boldsymbol{\epsilon}_t$ is used as the starting values for the GARCH constant parameters, and the starting value 10^{-6} is used for the other GARCH parameters. For the identification of the parameters of a *BEKK* representation GARCH model, the diagonal elements of the GARCH constant parameters and the first element of the ARCH and GARCH parameters are restricted to be positive.

Covariance Stationarity

Define the multivariate GARCH process as

$$\mathbf{h}_t = \sum_{i=1}^{\infty} G(B)^{i-1}[\mathbf{c} + A(B)\boldsymbol{\eta}_t]$$

where $\mathbf{h}_t = \text{vec}(H_t)$, $\mathbf{c} = \text{vec}(C_0'C_0)$, and $\boldsymbol{\eta}_t = \text{vec}(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t')$ This representation is equivalent to a GARCH(p,q) model by the following algebra:

$$\begin{aligned}
\mathbf{h}_t &= \mathbf{c} + A(B)\boldsymbol{\eta}_t + \sum_{i=2}^{\infty} G(B)^{i-1}[\mathbf{c} + A(B)\boldsymbol{\eta}_t] \\
&= \mathbf{c} + A(B)\boldsymbol{\eta}_t + G(B)\sum_{i=1}^{\infty} G(B)^{i-1}[\mathbf{c} + A(B)\boldsymbol{\eta}_t] \\
&= \mathbf{c} + A(B)\boldsymbol{\eta}_t + G(B)\mathbf{h}_t
\end{aligned}$$

Defining $A(B) = \sum_{i=1}^{q}(A_i \otimes A_i)'B^i$ and $G(B) = \sum_{i=1}^{p}(G_i \otimes G_i)'B^i$ gives a *BEKK* representation; defining $\mathbf{h}_t = \text{vech}(H_t)$, $\mathbf{c} = \mathbf{c}^*$, $\boldsymbol{\eta}_t = \text{vech}(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t')$, $A(B) = \sum_{i=1}^{q} A_i^* B^i$, and $G(B) = \sum_{i=1}^{p} G_i^* B^i$ gives other representations.

The necessary and sufficient conditions for covariance stationarity of the multivariate GARCH process is that all the eigenvalues of $A(1) + G(1)$ are less than one in modulus.

An Example of a VAR(1)-ARCH(1) Model

The following DATA steps simulate a bivariate vector time series to provide test data for the multivariate GARCH model:

```
data garch;
   retain seed 16587;
   esq1 = 0; esq2 = 0;
```

```
      ly1 = 0;  ly2 = 0;
      do i = 1 to 1000;
         ht = 6.25 + 0.5*esq1;
         call rannor(seed,ehat);
         e1 = sqrt(ht)*ehat;
         ht = 1.25 + 0.7*esq2;
         call rannor(seed,ehat);
         e2 = sqrt(ht)*ehat;
         y1 = 2 + 1.2*ly1 - 0.5*ly2 + e1;
         y2 = 4 + 0.6*ly1 + 0.3*ly2 + e2;
         if i>500 then output;
         esq1 = e1*e1; esq2 = e2*e2;
         ly1 = y1;  ly2 = y2;
      end;
      keep y1 y2;
   run;
```

First fit a VAR(1) model to the data.

```
   proc varmax data=garch;
      model y1 y2 / p=1;
   run;
```

```
                             The VARMAX Procedure

                                  Schematic
                              Representation of
                             Parameter Estimates

                            Variable/
                            Lag           C    AR1

                            y1            +    +-
                            y2            +    ++

                             + is > 2*std error,
                            - is < -2*std error,
                                . is between,
                                  * is N/A


                                 Information
                                   Criteria

                               AICC    3.708727
                               HQC     3.728459
                               AIC     3.708581
                               SBC     3.759234
                               FPEC     40.7959


                  Univariate Model White Noise Diagnostics

                    Durbin             Normality                 ARCH
Variable            Watson    Chi-Square     Pr > ChiSq    F Value    Pr > F

y1                 2.03478         26.36         <.0001     117.99    <.0001
y2                 1.69962        980.99         <.0001     580.90    <.0001
```

Figure 30.56. Parameter Estimates and Model Diagnostics for the VAR(1) Model

In Figure 30.56, you can see that the AR coefficients are significant. In the information criteria table, the AICC value is 3.7087. The normality and ARCH effect tests of residuals show that the fitted VAR(1) model is not good for the data.

The following statements fit a VAR(1)-ARCH(1) model to the data. For an AR-ARCH model, you specify the order of the autoregressive model with the P=1 option and the GARCH=(Q=1) option; you specify the quasi-Newton optimization in the NLOPTIONS statement as an optimization technique.

```
proc varmax data=garch;
   model y1 y2 / p=1 garch=(q=1);
   nloptions tech=qn;
run;
```

```
                              The VARMAX Procedure

                                   Schematic
                               Representation of
                              Parameter Estimates

                             Variable/
                             Lag           C    AR1

                             y1            +    +-
                             y2            +    ++

                              + is > 2*std error,
                             - is < -2*std error,
                                 . is between,
                                   * is N/A


                           Schematic Representation
                              of GARCH Parameter
                                   Estimates

                           Variable/
                           Lag           GCHC    ACH1

                           h1            +.      +.
                           h2            *+      --

                            + is > 2*std error,  -
                             is < -2*std error,  .
                             is between,  * is N/A


                                  Information
                                    Criteria

                                AICC    2.666632
                                HQC     2.686364
                                AIC     2.666486
                                SBC     2.717139
                                FPEC    14.38932
```

Figure 30.57. Parameter Estimates and Model Diagnostics for the VAR(1)-ARCH(1) Model

In Figure 30.57, you can see that the AR and ARCH coefficients are significant. The AICC value is 2.7402, so it is smaller than the AICC value in Figure 30.56. The VAR(1)-ARCH(1) model fits the data better than the VAR(1) model.

The following outputs show the details of this example.

```
                              The VARMAX Procedure

                               Optimization Start
                              Parameter Estimates
                                                               Gradient
                                                              Objective
                N Parameter         Estimate                   Function

                1 CONST1            2.249575                  -0.012507
                2 CONST2            3.902673                  -0.050279
                3 AR1_1_1           1.231775                   0.433349
                4 AR1_2_1           0.576890                   0.507002
                5 AR1_1_2          -0.528405                   0.011324
                6 AR1_2_2           0.343714                  -0.147973
                7 GCHC1_1           3.162082                 483.349603
                8 GCHC1_2           0.069701                  -0.065497
                9 GCHC2_2           2.014644                 464.020185
               10 ACH1_1_1       0.000001054                   8.056916
               11 ACH1_2_1       0.000001054                   0.207431
               12 ACH1_1_2       0.000001054                  -0.281331
               13 ACH1_2_2       0.000001054                  -0.017410
```

Figure 30.58. Start Parameter Estimates for the VAR(1)-ARCH(1) Model

Figure 30.58 shows the initial values of parameters. GCHC1_1, GCHC2_2, and ACH1_1_1 are the log transformed values of their initial values.

```
                              The VARMAX Procedure

                              Optimization Results
                              Parameter Estimates
                                                               Gradient
                                                              Objective
                N Parameter         Estimate                   Function

                1 CONST1            2.147471                   0.000382
                2 CONST2            4.144038                  -0.000220
                3 AR1_1_1           1.214502                  -0.007029
                4 AR1_2_1           0.607698                  -0.002985
                5 AR1_1_2          -0.521841                  -0.003333
                6 AR1_2_2           0.298631                  -0.001484
                7 GCHC1_1           1.049782               -0.000046162
                8 GCHC1_2          -0.060821                   0.001307
                9 GCHC2_2           0.212161                   0.000453
               10 ACH1_1_1         -0.993632                  -0.000355
               11 ACH1_2_1         -0.200118                  -0.000997
               12 ACH1_1_2         -0.044265                   0.002379
               13 ACH1_2_2         -0.728550                0.000080949
```

Figure 30.59. Results of Parameter Estimates for the VAR(1)-ARCH(1) Model

Figure 30.59 shows the final parameter estimates. GCHC1_1, GCHC2_2, and ACH1_1_1 are the log transformed estimates.

```
                          The VARMAX Procedure

              Type of Model                             VAR(1)-ARCH(1)
              Estimation Method      Maximum Likelihood Estimation
              Representation Type                                BEKK


                        GARCH Constant Estimates

                    Variable               h1              h2

                    h1                2.85703        -0.06082
                    h2                 .              1.23635


                        ARCH Coefficients Matrix

                  Lag    Variable                 e1               e2

                    1    h1                  0.37023         -0.04426
                         h2                 -0.20012         -0.72855
```

Figure 30.60. Parameter Estimates for the VAR(1)-ARCH(1) Model

Figure 30.60 shows that the conditional variance fits the *BEKK* representation ARCH(1) model.

$$\begin{aligned}
\boldsymbol{\epsilon}_t|\mathcal{F}(t-1) &\sim N(0,H_t) \\
H_t &= \begin{bmatrix} 8.3634 & -0.1534 \\ -0.1534 & 1.6011 \end{bmatrix} \\
&+ \begin{bmatrix} 0.3707 & 0.0532 \\ 0.0287 & 0.7121 \end{bmatrix}' \boldsymbol{\epsilon}_{t-1}\boldsymbol{\epsilon}'_{t-1} \begin{bmatrix} 0.3707 & 0.0532 \\ 0.0287 & 0.7121 \end{bmatrix}
\end{aligned}$$

Since $C_0 = \begin{bmatrix} 2.8920 & -0.0531 \\ 0 & 1.2643 \end{bmatrix}$ is an upper triangular matrix, the constant term of the ARCH model is obtained by $C_0'C_0 = \begin{bmatrix} 8.3634 & -0.1534 \\ -0.1534 & 1.6011 \end{bmatrix}$.

```
                              The VARMAX Procedure

                            Model Parameter Estimates

                                            Standard
Equation Parameter       Estimate          Error t Value Pr > |t| Variable

y1       CONST1           2.14747      0.20622   10.41   0.0001 1
         AR1_1_1          1.21450      0.02548   47.66   0.0001 y1(t-1)
         AR1_1_2         -0.52184      0.02856  -18.27   0.0001 y2(t-1)
y2       CONST2           4.14404      0.09396   44.11   0.0001 1
         AR1_2_1          0.60770      0.01091   55.71   0.0001 y1(t-1)
         AR1_2_2          0.29863      0.01347   22.17   0.0001 y2(t-1)


                         GARCH Model Parameter Estimates

                                              Standard
     Parameter          Estimate                Error     t Value    Pr > |t|

     GCHC1_1             2.85703              0.03922       72.85      0.0001
     GCHC1_2            -0.06082              0.07144       -0.85      0.3950
     GCHC2_2             1.23635              0.05309       23.29      0.0001
     ACH1_1_1            0.37023              0.16365        2.26      0.0241
     ACH1_2_1           -0.20012              0.06988       -2.86      0.0044
     ACH1_1_2           -0.04426              0.02332       -1.90      0.0582
     ACH1_2_2           -0.72855              0.05950      -12.25      0.0001
```

Figure 30.61. Parameter Estimates for the VAR(1)-ARCH(1) Model

Figure 30.61 shows the AR and ARCH parameter estimates and their significance. The ARCH parameters are estimated by the vectorized parameter matrices.

The fitted VAR(1) model with the previous conditional covariance ARCH models is written as follows:

$$\mathbf{y}_t = \begin{bmatrix} 2.1565 \\ 4.0487 \end{bmatrix} + \begin{bmatrix} 1.2215 & -0.5317 \\ 0.6079 & 0.3040 \end{bmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

```
                          The VARMAX Procedure

                   Roots of AR Characteristic Polynomial

Index         Real      Imaginary      Modulus       Radian       Degree

    1      0.75657        0.32775       0.8245       0.4088      23.4223
    2      0.75657       -0.32775       0.8245      -0.4088     -23.4223


                  Roots of GARCH Characteristic Polynomial

Index         Real      Imaginary      Modulus       Radian       Degree

    1      0.54251        0.00000       0.5425       0.0000       0.0000
    2      0.14306        0.00000       0.1431       0.0000       0.0000
    3     -0.27859        0.00000       0.2786       3.1416     180.0000
    4     -0.27859        0.00000       0.2786       3.1416     180.0000
```

Figure 30.62. Roots for the VAR(1)-ARCH(1) Model

Figure 30.62 shows the roots of the AR and ARCH characteristic polynomials; the eigenvalues have a modulus less than one.

Add the OUTPUT statement in the previous example. The LEAD= and BACK= options are ignored when you fit the GARCH-type models. The created data set, FOR, includes the predicted values of the dependent variables within samples.

```
proc varmax data=garch;
   model y1 y2 / p=1 garch=(q=1);
   nloptions tech=qn;
   output out=for lead=5 back=3;
run;
```

OUT= Data Set

The OUT= data set contains the forecast values produced by the OUTPUT statement. The following output variables can be created:

- the BY variables
- the ID variable
- the MODEL statement dependent (endogenous) variables. These variables contain the actual values from the input data set.
- FOR*i*, numeric variables containing the forecasts. The FOR*i* variables contain the forecasts for the *i*th endogenous variable in the MODEL statement list. Forecasts are 1-step-ahead predictions until the end of the data or until the observation specified by the BACK= option.
- RES*i*, numeric variables containing the residual for the forecast of the *i*th endogenous variable in the MODEL statement list. For forecast observations, the actual values are missing and the RES*i* variables contain missing values.

- STD*i*, numeric variables containing the standard deviation for the forecast of the *i*th endogenous variable in the MODEL statement list. The values of the STD*i* variables can be used to construct univariate confidence limits for the corresponding forecasts.
- LCI*i*, numeric variables containing the lower confidence limits for the corresponding forecasts of the *i*th endogenous variable in the MODEL statement list.
- UCI*i*, numeric variables containing the upper confidence limits for the corresponding forecasts of the *i*th endogenous variable in the MODEL statement list.

Table 30.2. OUT= Data Set

Obs	ID variable	y1	FOR1	RES1	STD1	LCI1	UCI1
1	date	y_{11}	f_{11}	r_{11}	σ_{11}	l_{11}	u_{11}
2	date	y_{12}	f_{12}	r_{12}	σ_{11}	l_{12}	u_{12}
⋮							

Obs	y2	FOR2	RES2	STD2	LCI2	UCI2
1	y_{21}	f_{21}	r_{21}	σ_{22}	l_{21}	u_{21}
2	y_{22}	f_{22}	r_{22}	σ_{22}	l_{22}	u_{22}
⋮						

The OUT= data set contains the values shown in Table 30.2 for a bivariate case.

Consider the following example:

```
proc varmax data=simul1 noprint;
   id date interval=year;
   model y1 y2 / p=1 noint;
   output out=out lead=5;
run;
proc print data=out(firstobs=98);
run;
```

```
Obs   date      y1         FOR1        RES1        STD1       LCI1        UCI1

 98   1997   -0.58433   -0.13500   -0.44934   1.13523   -2.36001    2.09002
 99   1998   -2.07170   -1.00649   -1.06522   1.13523   -3.23150    1.21853
100   1999   -3.38342   -2.58612   -0.79730   1.13523   -4.81113   -0.36111
101   2000     .        -3.59212     .        1.13523   -5.81713   -1.36711
102   2001     .        -3.09448     .        1.70915   -6.44435    0.25539
103   2002     .        -2.17433     .        2.14472   -6.37792    2.02925
104   2003     .        -1.11395     .        2.43166   -5.87992    3.65203
105   2004     .        -0.14342     .        2.58740   -5.21463    4.92779

Obs      y2         FOR2        RES2       STD2       LCI2        UCI2

 98    0.64397   -0.34932    0.99329   1.19096   -2.68357    1.98492
 99    0.35925   -0.07132    0.43057   1.19096   -2.40557    2.26292
100   -0.64999   -0.99354    0.34355   1.19096   -3.32779    1.34070
101     .        -2.09873     .        1.19096   -4.43298    0.23551
102     .        -2.77050     .        1.47666   -5.66469    0.12369
103     .        -2.75724     .        1.74212   -6.17173    0.65725
104     .        -2.24943     .        2.01925   -6.20709    1.70823
105     .        -1.47460     .        2.25169   -5.88782    2.93863
```

Figure 30.63. OUT= Data Set

The output in Figure 30.63 shows the part of the results of the OUT= data set.

OUTEST= Data Set

The OUTEST= data set contains estimation results of the fitted model. The following output variables can be created:

- the BY variables
- NAME, a character variable containing the name of endogenous (dependent) variables or the name of the parameters for the covariance of the matrix of the parameter estimates if the OUTCOV option is specified
- TYPE, a character variable containing the value EST for parameter estimates, the value STD for standard error of parameter estimates, and the value COV for the covariance of the matrix of the parameter estimates if the OUTCOV option is specified
- CONST, a numeric variable containing the estimates of constant parameters and their standard errors
- SEASON_i, a numeric variable containing the estimates of seasonal dummy parameters and their standard errors, where $i = 1, \ldots, (nseason - 1)$
- LTREND, a numeric variable containing the estimates of linear trend parameters and their standard errors
- QTREND, a numeric variable containing the estimates of quadratic trend parameters and their standard errors
- XLl_i, numeric variables containing the estimates of exogenous parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, r$, where r is the number of exogenous variables

- ARl_i, numeric variables containing the estimates of autoregressive parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$, where k is the number of endogenous variables
- MAl_i, numeric variables containing the estimates of moving-average parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$, where k is the number of endogenous variables
- GCHM_i are numeric variables containing the estimates of the GARCH-M parameters of the covariance matrix and their standard errors, where $i = 1, \ldots, k(k+1)/2$, k is the number of endogenous variables.
- ACHl_i are numeric variables containing the estimates of the ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for a *BEKK* representation, or $i = 1, \ldots, k(k+1)/2$ for other representations. k is the number of endogenous variables.
- GCHl_i are numeric variables containing the estimates of the GARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for a *BEKK* representation, or $i = 1, \ldots, k(k+1)/2$ for other representations. k is the number of endogenous variables.
- GCHC_i are numeric variables containing the estimates of the constant parameters of the covariance matrix and their standard errors, where $i = 1, \ldots, k$ for a *BEKK* representation, k is the number of endogenous variables. $i = 1$ for other representations.

Table 30.3. OUTEST= Data Set

Obs	NAME	TYPE	CONST	AR1_1	AR1_2	AR2_1	AR2_2
1	y1	EST	δ_1	$\phi_{1,11}$	$\phi_{1,12}$	$\phi_{2,11}$	$\phi_{2,12}$
2		STD	se(δ_1)	se($\phi_{1,11}$)	se($\phi_{1,12}$)	se($\phi_{2,11}$)	se($\phi_{2,12}$)
3	y2	EST	δ_2	$\phi_{1,21}$	$\phi_{1,22}$	$\phi_{2,21}$	$\phi_{2,22}$
4		STD	se(δ_2)	se($\phi_{1,21}$)	se($\phi_{1,22}$)	se($\phi_{2,21}$)	se($\phi_{2,22}$)

The OUTEST= data set contains the values shown Table 30.3 for a bivariate case.

Consider the following example:

```
proc varmax data=simul2 outest=est;
   model y1 y2 / p=2 noint noprint
                 ecm=(rank=1 normalize=y1);
run;
proc print data=est;
run;
```

```
     Obs     NAME     TYPE       AR1_1         AR1_2         AR2_1         AR2_2

      1       y1      EST      -0.46680       0.91295      -0.74332      -0.74621
      2               STD       0.04786       0.09359       0.04526       0.04769
      3       y2      EST       0.10667      -0.20862       0.40493      -0.57157
      4               STD       0.05146       0.10064       0.04867       0.05128
```

Figure 30.64. OUTEST= Data Set

The output in Figure 30.64 shows the part of results of the OUTEST= data set.

OUTSTAT= Data Set

The OUTSTAT= data set contains estimation results of the fitted model. The following output variables can be created. The sub-index i is $1, \ldots, k$, where k is the number of endogenous variables.

- the BY variables
- NAME, a character variable containing the name of endogenous (dependent) variables
- SIGMA_i, numeric variables containing the estimate covariance of the innovation covariance matrix
- AICC, a numeric variable containing the corrected Akaike's information criteria value
- RSquare, a numeric variable containing R^2
- FValue, a numeric variable containing the F statistics
- PValue, a numeric variable containing p-value for the F statistics

If the JOHANSEN= option is specified, the following items are added:

- Eigenvalue, a numeric variable containing eigenvalues for the cointegration rank test of integrated order 1
- RestrictedEigenvalue, a numeric variable containing eigenvalues for the cointegration rank test of integrated order 1 when the NOINT option is not specified
- Beta_i, numeric variables containing $\boldsymbol{\beta}$ long-run effect parameter estimates
- Alpha_i, numeric variables containing $\boldsymbol{\alpha}$ adjustment parameter estimates

If the JOHANSEN=(IORDER=2) option is specified, the following items are added:

- EValueI2_i, numeric variables containing eigenvalues for the cointegration rank test of integrated order 2
- EValueI1, a numeric variable containing eigenvalues for the cointegration rank test of integrated order 1

- Eta_i, numeric variables containing $\boldsymbol{\eta}$ parameter estimates in integrated order 2
- Xi_i, numeric variables containing $\boldsymbol{\xi}$ parameter estimates in integrated order 2

Table 30.4. OUTSTAT= Data Set

Obs	NAME	SIGMA_1	SIGMA_2	AICC	RSquare	FValue	PValue
1	y1	σ_{11}	σ_{12}	$aicc$	R_1^2	F_1	$prob_1$
2	y2	σ_{21}	σ_{22}	.	R_2^2	F_2	$prob_2$

Obs	EValueI2_1	EValueI2_2	EValueI1	Beta_1	Beta_2
1	e_{11}	e_{12}	e_1	β_{11}	β_{12}
2	e_{21}	.	e_2	β_{21}	β_{21}

Obs	Alpha_1	Alpha_2	Eta_1	Eta_2	Xi_1	Xi_2
1	α_{11}	α_{12}	η_{11}	η_{12}	ξ_{11}	ξ_{12}
2	α_{21}	α_{22}	η_{21}	η_{22}	ξ_{21}	ξ_{22}

The OUTSTAT= data set contains the values shown Table 30.4 for a bivariate case.

Consider the following example:

```
proc varmax data=simul2 outstat=stat;
   model y1 y2 / p=2 noint
                 cointtest=(johansen=(iorder=2))
                 ecm=(rank=1 normalize=y1) noprint;
run;
proc print data=stat;
run;
```

```
                                                            E       E
                                                            V       V
                                                            a       a       E
         S       S               R                          l       l       V
         I       I               S       F         P        u       u       a
         G       G               q       V         V        e       e       l
  N      M       M       A       u       a         a        I       I       u
O A      A       A       I       a       l         l        2       2       e
b M      _       _       C       r       u         u        _       _       I
s E      1       2       C       e       e         e        1       2       1

1 y1 94.7557   4.527 9.37221 0.93905 482.782 5.9027E-57 0.98486 0.95079 0.50864
2 y2  4.5268 109.570  .      0.94085 498.423 1.4445E-57 0.81451  .      0.01108




                         A        A
       B        B        l        l
       e        e        p        p        E         E
       t        t        h        h        t         t        X        X
O      a        a        a        a        a         a        i        i
b      _        _        _        _        _         _        _        _
s      1        2        1        2        1         2        1        2

1  1.00000  1.00000 -0.46680 0.007937 -0.012307 0.027030  54.1606 -52.3144
2 -1.95575 -1.33622  0.10667 0.033530  0.015555 0.023086 -79.4240 -18.3308
```

Figure 30.65. OUTSTAT= Data Set

The output in Figure 30.65 shows the part of results of the OUTSTAT= data set.

Printed Output

The default printed output produced by the VARMAX procedure is described in the following list:

- descriptive statistics, which include the number of observations used, the names of the variables, their means and standard deviations (STD), their minimums and maximums, the differencing operations used, and the labels of the variables
- a type of model to fit the data and an estimate method
- the estimates of the constant vector (or seasonal constant matrix), the trend vector, the coefficients matrices of the distributed lags, the AR coefficients matrices, and the MA coefficients matrices
- a table of parameter estimates showing the following for each parameter: the variable name for the left-hand side of equations, the parameter name, the parameter estimate, the approximate standard error, *t* value, the approximate probability ($Pr > |t|$), and the variable name for the right-hand side of equations in terms of each parameter
- the innovation covariance matrix
- the Information criteria

- the cross-covariance and cross-correlation matrices of the residuals
- the tables of test statistics for the hypothesis that the residuals of the model are white noise:
  - Durbin-Watson (DW) statistics
  - F test for autoregressive conditional heteroscedastic (ARCH) disturbances
  - F test for AR disturbance
  - Jarque-Bera normality test
  - Portmanteau test

ODS Table Names

The VARMAX procedure assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table:

Table 30.5. ODS Tables Produced in the VARMAX Procedure

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
AccumImpulse	Accumulated Impulse Response Matrices	IMPULSE=(ACCUM) IMPULSE=(ALL)
AccumImpulsebyVar	Accumulated Impulse Response by Variable	IMPULSE=(ACCUM) IMPULSE=(ALL)
AccumImpulseX	Accumulated Transfer Function Matrices	IMPULSX=(ACCUM) IMPULSX=(ALL)
AccumImpulseXbyVar	Accumulated Transfer Function by Variable	IMPULSX=(ACCUM) IMPULSX=(ALL)
Alpha	α Coefficients	JOHANSEN=
AlphaInECM	α Coefficients When Rank=r	ECM=
AlphaOnDrift	α Coefficients under the Restriction of a Deterministic Term	JOHANSEN=
AlphaBetaInECM	$\Pi = \alpha\beta'$ Coefficients When Rank=r	ECM=
ANOVA	Univariate Model Diagnostic Checks for the Residuals	default
ARCHCoef	ARCH Coefficients	GARCH=
ARCoef	AR Coefficients	P=
ARRoots	Roots of AR Characteristic Polynomial	ROOTS with P=
Beta	β Coefficients	JOHANSEN=
BetaInECM	β Coefficients When Rank=r	ECM=
BetaOnDrift	β Coefficients under the Restriction of a Deterministic Term	JOHANSEN=
Constant	Constant Estimates	w/o NOINT
CorrB	Correlations of Parameter Estimates	CORRB
CorrResiduals	Correlations of Residuals	default

Table 30.5. (ODS Tables Continued)

ODS Table Name	Description	Option
CorrResidualsbyVar	Correlations of Residuals by Variable	default
CorrResidualsGraph	Schematic Representation of Correlations of Residuals	default
CorrXGraph	Schematic Representation of Sample Correlations of Independent Series	CORRX
CorrYGraph	Schematic Representation of Sample Correlations of Dependent Series	CORRY
CorrXLags	Correlations of Independent Series	CORRX
CorrXbyVar	Correlations of Independent Series by Variable	CORRX
CorrYLags	Correlations of Dependent Series	CORRY
CorrYbyVar	Correlations of Dependent Series by Variable	CORRY
CovB	Covariances of Parameter Estimates	COVB
CovInnovation	Covariances of the Innovations	default
CovPredictError	Covariance Matrices of the Prediction Error	COVPE
CovPredictErrorbyVar	Covariances of the Prediction Error by Variable	COVPE
CovResiduals	Covariances of Residuals	default
CovResidualsbyVar	Covariances of Residuals by Variable	default
CovXLags	Covariances of Independent Series	COVX
CovXbyVar	Covariances of Independent Series by Variable	COVX
CovYLags	Covariances of Dependent Series	COVY
CovYbyVar	Covariances of Dependent Series by Variable	COVY
DecomposeCov-PredictError	Decomposition of the Prediction Error Covariances	DECOMPOSE
DecomposeCov-PredictErrorbyVar	Decomposition of the Prediction Error Covariances by Variable	DECOMPOSE
DFTest	Dickey-Fuller Test	DFTEST
DiagnostAR	Test the AR Disturbance for the Residuals	default
DiagnostWN	Test the ARCH Disturbance and Normality for the Residuals	default
DynamicARCoef	AR Coefficients of the Dynamic Model	DYNAMIC
DynamicConstant	Constant Estimates of the Dynamic Model	DYNAMIC
DynamicCov-Innovation	Covariances of the Innovations of the Dynamic Model	DYNAMIC
DynamicLinearTrend	Linear Trend Estimates of the Dynamic Model	DYNAMIC
DynamicMACoef	MA Coefficients of the Dynamic Model	DYNAMIC

Table 30.5. (ODS Tables Continued)

ODS Table Name	Description	Option
DynamicSConstant	Seasonal Constant Estimates of the Dynamic Model	DYNAMIC
DynamicParameter-Estimates	Parameter Estimates Table of the Dynamic Model	DYNAMIC
DynamicParameter-Graph	Schematic Representation of the Parameters of the Dynamic Model	DYNAMIC
DynamicQuadTrend	Quadratic Trend Estimates of the Dynamic Model	DYNAMIC
DynamicSeasonGraph	Schematic Representation of the Seasonal Dummies of the Dynamic Model	DYNAMIC
DynamicXLagCoef	Dependent Coefficients of the Dynamic Model	DYNAMIC
Hypothesis	Hypothesis of Different Deterministic Terms in Cointegration Rank Test	JOHANSEN=
HypothesisTest	Test Hypothesis of Different Deterministic Terms in Cointegration Rank Test	JOHANSEN=
EigenvalueI2	Eigenvalues in Integrated Order 2	JOHANSEN= (IORDER=2)
Eta	η Coefficients	JOHANSEN= (IORDER=2)
GARCHConstant	GARCH Constant Estimates	GARCH=
GARCHParameter-Estimates	GARCH Parameter Estimates Table	GARCH=
GARCHParameter-Graph	Schematic Representation of the Garch Parameters	GARCH=
GARCHRoots	Roots of GARCH Characteristic Polynomial	ROOTS with GARCH=
InfiniteARRepresent	Infinite Order AR Representation	IARR
InfoCriteria	Information criteria	default
LinearTrend	Linear Trend Estimates	TREND=
MACoef	MA Coefficients	Q=
MARoots	Roots of MA Characteristic Polynomial	ROOTS with Q=
MaxTest	Cointegration Rank Test Using the Maximum Eigenvalue	JOHANSEN= (TYPE=MAX)
Minic	Tentative Order Selection	MINIC MINIC=
ModelType	Type of Model	default
NObs	Number of Observations	default
OrthoImpulse	Orthogonalized Impulse Response Matrices	IMPULSE=(ORTH) IMPULSE=(ALL)
OrthoImpulsebyVar	Orthogonalized Impulse Response by Variable	IMPULSE=(ORTH) IMPULSE=(ALL)
ParameterEstimates	Parameter Estimates Table	default

Table 30.5. (ODS Tables Continued)

ODS Table Name	Description	Option
ParameterGraph	Schematic Representation of the Parameters	default
PartialAR	Partial Autoregression Matrices	PARCOEF
PartialARGraph	Schematic Representation of Partial Autoregression	PARCOEF
PartialCanCorr	Partial Canonical Correlation Analysis	PCANCORR
PartialCorr	Partial Cross-Correlation Matrices	PCORR
PartialCorrbyVar	Partial Cross-Correlation by Variable	PCORR
PartialCorrGraph	Schematic Representation of Partial Cross-Correlations	PCORR
PortmanteauTest	Chi-Square Test Table for Residual Cross-Correlations	default
ProportionCov-PredictError	Proportions of Prediction Error Covariance Decomposition	DECOMPOSE
ProportionCov-PredictErrorbyVar	Proportions of Prediction Error Covariance Decomposition by Variable	DECOMPOSE
RankTestI2	Cointegration Rank Test in Integrated Order 2	JOHANSEN= (IORDER=2)
RestrictMaxTest	Cointegration Rank Test Using the Maximum Eigenvalue under the Restriction of a Deterministic Term	JOHANSEN= (TYPE=MAX) w/o NOINT
RestrictTraceTest	Cointegration Rank Test Using the Trace under the Restriction of a Deterministic Term	JOHANSEN= (TYPE=TRACE) w/o NOINT
QuadTrend	Quadratic Trend Estimates	TREND=QUAD
SeasonGraph	Schematic Representation of the Seasonal Dummies	default
SConstant	Seasonal Constant Estimates	NSEASON=
SimpleImpulse	Impulse Response Matrices	IMPULSE IMPULSE=(SIMPLE) IMPULSE=(ALL)
SimpleImpulsebyVar	Impulse Response by Variable	IMPULSE IMPULSE=(SIMPLE) IMPULSE=(ALL)
SimpleImpulseX	Impulse Response Matrices of Transfer Function	IMPULSX IMPULSX=(SIMPLE) IMPULSX=(ALL)
SimpleImpulseXbyVar	Impulse Response of Transfer Function by Variable	IMPULSX IMPULSX=(SIMPLE) IMPULSX=(ALL)
Summary	Simple Summary Statistics	default
SWTest	Common Trends Test	SW SW=
TraceTest	Cointegration Rank Test Using the Trace	JOHANSEN= (TYPE=TRACE)

Table 30.5. (ODS Tables Continued)

ODS Table Name	Description	Option
Xi	ξ Coefficient Matrix	JOHANSEN= (IORDER=2)
XLagCoef	Dependent Coefficients	XLAG=
YWEstimates	Yule-Walker Estimates	YW
ODS Tables Created by the COINTEG Statement		
AlphaInECM	α Coefficients When Rank=r	default
AlphaBetaInECM	$\Pi = \alpha\beta'$ Coefficients When Rank=r	default
AlphaOnAlpha	α Coefficients under the Restriction of α	J=
AlphaOnBeta	α Coefficients under the Restriction of β	H=
AlphaTestResult	Hypothesis Testing of β	J=
BetaInECM	β Coefficients When Rank=r	default
BetaOnBeta	β Coefficients under the Restriction of β	H=
BetaOnAlpha	β Coefficients under the Restriction of α	J=
BetaTestResult	Hypothesis Testing of β	H=
HMatrix	Restriction Matrix for β	H=
JMatrix	Restriction Matrix for α	J=
WeakExogeneity	Testing Weak Exogeneity of each Dependent Variable with Respect to BETA	EXOGENEITY
ODS Tables Created by the CAUSAL Statement		
CausalityTest	Granger-Causality Test	default
GroupVars	Two Groups of Variables	default
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction Table	default
ODS Tables Created by the TEST Statement		
Test	Wald Test	default
ODS Tables Created by the OUTPUT Statement		
Forecasts	Forecasts Table	w/o NOPRINT

Note that the ODS table names suffixed by “byVar” can obtained with the PRINTFORM=UNIVAIRTE option.

ODS Graphics (Experimental)

This section describes the use of ODS for creating statistical graphs with the VARMAX procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in the future release.

To request these graphs you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

When the ODS GRAPHICS are in effect, the VARMAX procedure produces a variety of plots for each dependent variable.

The plots available are as follows:

- The procedure displays the following plots for each dependent variable in the MODEL statement:
  - Time series and predicted series
  - Residual series
  - Histogram of the residuals
  - ACF of the residuals
  - PACF of the residuals
  - White noise test of the residuals
- The procedure displays forecast plots for each dependent variable in the OUTPUT statement.

ODS Graph Names

PROC VARMAX assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 30.6.

To request these graphs you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, "Statistical Graphics Using ODS."

Table 30.6. ODS Graphics Produced in the VARMAX Procedure

ODS Graph Name	Plot Description	Statement
ErrorACFPlot#	Autocorrelation Function of Residuals	Model
ErrorHistogram#	Histogram of Residuals	Model
ErrorPACFPlot#	Partial Autocorrelation Function of Residuals	Model
ErrorPlot#	Residuals	Model
ErrorWhiteNoisePlot#	White Noise Test of Residuals	Model
ForecastsOnlyPlot#	Forecasts	OUTPUT
ModelForecastsPlot#	Time Series and Forecasts	OUTPUT
ModelPlot#	Time Series and Predicted Series	Model

Note that the symbol “#” that follows the ODS graphics names is the number corresponding to the order in which the dependent variable appears in the MODEL statement.

Examples

Example 30.1. Analysis of U.S. Economic Variables

Consider the following four-dimensional system of U.S. economic variables. Quarterly data for the years 1954 to 1987 are used (Lütkepohl 1993, Table E.3.). The following statements plot the series and proceed with the VARMAX procedure.

```
symbol1 v=none height=1 c=black;
symbol2 v=none height=1 c=black;

   title 'Analysis of U.S. Economic Variables';
   data us_money;
      date=intnx( 'qtr', '01jan54'd, _n_-1 );
      format date yyq. ;
      input y1 y2 y3 y4 @@;
      y1=log(y1);
      y2=log(y2);
      label y1='log(real money stock M1)'
            y2='log(GNP in bil. of 1982 dollars)'
            y3='Discount rate on 91-day T-bills'
            y4='Yield on 20-year Treasury bonds';
      datalines;
      ... data lines omitted ...
      ;

   legend1 across=1 frame label=none;

   proc gplot data=us_money;
      symbol1 i = join l = 1;
      symbol2 i = join l = 2;
      axis2 label = (a=-90 r=90 " ");
      plot y1 * date = 1 y2 * date = 2 /
           overlay vaxis=axis2 legend=legend1;
   run;

   proc gplot data=us_money;
      symbol1 i = join l = 1;
      symbol2 i = join l = 2;
      axis2 label = (a=-90 r=90 " ");
      plot y3 * date = 1 y4 * date = 2 /
           overlay vaxis=axis2 legend=legend1;
   run;

   proc varmax data=us_money;
      id date interval=qtr;
      model y1-y4 / p=2 lagmax=6 dftest
                    print=(iarr(3))
```

```
                     cointtest=(johansen=(iorder=2))
                     ecm=(rank=1 normalize=y1);
      cointeg rank=1 normalize=y1 exogeneity;
   run;
```

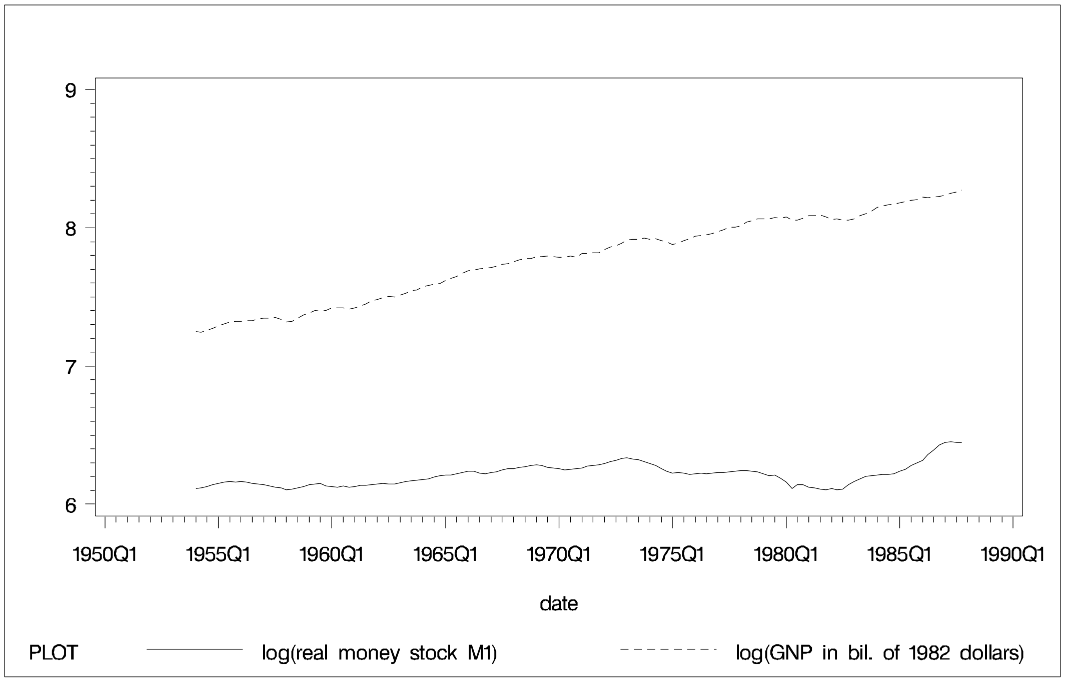
This example performs the Dickey-Fuller test for stationarity, the Johansen cointegrated test integrated order 2, and the exogeneity test. The VECM(2) fits the data. From the outputs shown below, you can see that the series has unit roots and is cointegrated in rank 1 with integrated order 1. The fitted VECM(2) is given as

$$\begin{aligned}\Delta \mathbf{y}_t &= \begin{pmatrix} 0.0408 \\ 0.0860 \\ 0.0052 \\ -0.0144 \end{pmatrix} + \begin{pmatrix} -0.0140 & 0.0065 & -0.2026 & 0.1306 \\ -0.0281 & 0.0131 & -0.4080 & 0.2630 \\ -0.0022 & 0.0010 & -0.0312 & 0.0201 \\ 0.0051 & -0.0024 & 0.0741 & -0.0477 \end{pmatrix} \mathbf{y}_{t-1} \\ &+ \begin{pmatrix} 0.3460 & 0.0913 & -0.3535 & -0.9690 \\ 0.0994 & 0.0379 & 0.2390 & 0.2866 \\ 0.1812 & 0.0786 & 0.0223 & 0.4051 \\ 0.0322 & 0.0496 & -0.0329 & 0.1857 \end{pmatrix} \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t\end{aligned}$$

The Δ prefixed to a variable name implies differencing.

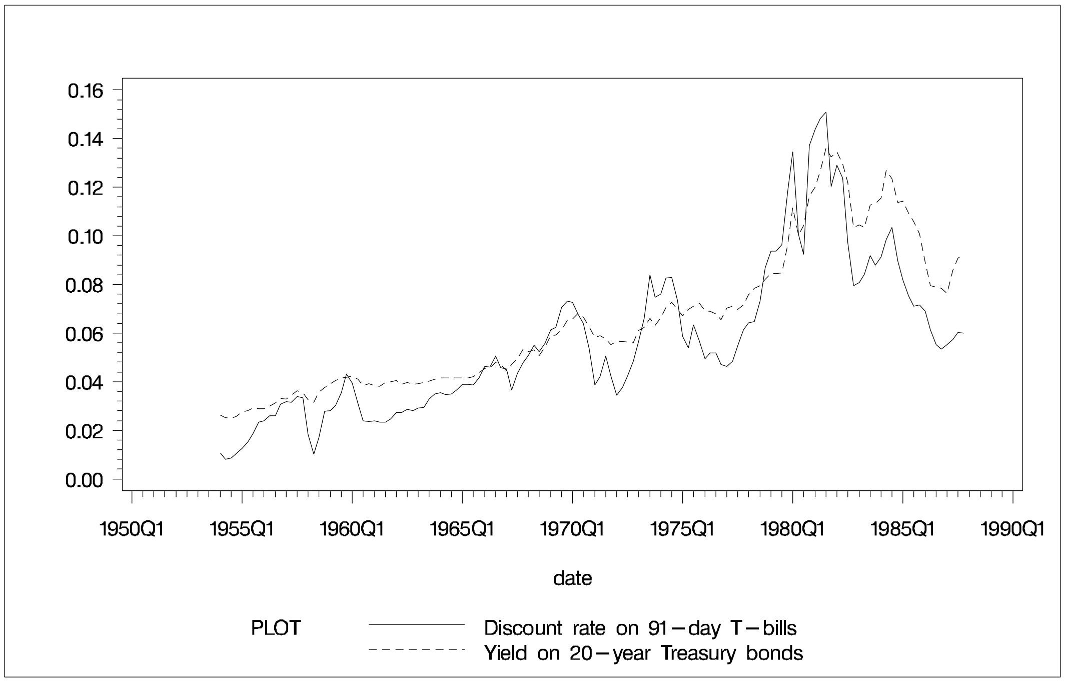
The following outputs show the details.

Output 30.1.1. Plot of Data



Output 30.1.1 shows the plot of the variables $y1$ and $y2$.

Output 30.1.2. Plot of Data Continued



Output 30.1.2 shows the plot of the variables $y3$ and $y4$.

Output 30.1.3. Descriptive Statistics

```
                                 The VARMAX Procedure

                          Number of Observations        136
                          Number of Pairwise Missing      0


                              Simple Summary Statistics

                                                   Standard
Variable Type            N          Mean     Deviation           Min           Max

y1       Dependent     136       6.21295       0.07924       6.10278       6.45331
y2       Dependent     136       7.77890       0.30110       7.24508       8.27461
y3       Dependent     136       0.05608       0.03109       0.00813       0.15087
y4       Dependent     136       0.06458       0.02927       0.02490       0.13600

                              Simple Summary Statistics

                    Variable Label

                    y1       log(real money stock M1)
                    y2       log(GNP in bil. of 1982 dollars)
                    y3       Discount rate on 91-day T-bills
                    y4       Yield on 20-year Treasury bonds
```

Output 30.1.3 shows the descriptive statistics.

Output 30.1.4. Unit Root Tests

```
                              The VARMAX Procedure

                         Dickey-Fuller Unit Root Tests

     Variable     Type              Rho     Pr < Rho         Tau     Pr < Tau

     y1           Zero Mean        0.05       0.6934        1.14       0.9343
                  Single Mean     -2.97       0.6572       -0.76       0.8260
                  Trend           -5.91       0.7454       -1.34       0.8725
     y2           Zero Mean        0.13       0.7124        5.14       0.9999
                  Single Mean     -0.43       0.9309       -0.79       0.8176
                  Trend           -9.21       0.4787       -2.16       0.5063
     y3           Zero Mean       -1.28       0.4255       -0.69       0.4182
                  Single Mean     -8.86       0.1700       -2.27       0.1842
                  Trend          -18.97       0.0742       -2.86       0.1803
     y4           Zero Mean        0.40       0.7803        0.45       0.8100
                  Single Mean     -2.79       0.6790       -1.29       0.6328
                  Trend          -12.12       0.2923       -2.33       0.4170
```

Output 30.1.4 shows the output for Dickey-Fuller tests for the nonstationarity of each series.

Output 30.1.5. Cointegration Rank Test

```
                                The VARMAX Procedure

                          Cointegration Rank Test for I(2)

                                                                            Trace
  r\k-r-s                4             3             2             1     of I(1)

       0         384.60903     214.37904     107.93782      37.02523     55.9633
       1                       219.62395      89.21508      27.32609     20.6542
       2                                      73.61779      22.13279      2.6477
       3                                                    38.29435      0.0149
5% CV I(2)        47.21000      29.38000      15.34000       3.84000

                                Cointegration Rank
                                  Test for I(2)

                                             5% CV of
                                r\k-r-s        I(1)

                                    0           47.21
                                    1           29.38
                                    2           15.34
                                    3            3.84
                              5% CV I(2)


                          Long-Run Parameter Beta Estimates

    Variable              1               2               3               4

    y1              1.00000         1.00000         1.00000         1.00000
    y2             -0.46458        -0.63174        -0.69996        -0.16140
    y3             14.51619        -1.29864         1.37007        -0.61806
    y4             -9.35520         7.53672         2.47901         1.43731


                       Adjustment Coefficient Alpha Estimates

    Variable              1               2               3               4

    y1             -0.01396         0.01396        -0.01119         0.00008
    y2             -0.02811        -0.02739        -0.00032         0.00076
    y3             -0.00215        -0.04967        -0.00183        -0.00072
    y4              0.00510        -0.02514        -0.00220         0.00016


                              Parameter Eta Estimates

    Variable              1               2               3               4

    y1             52.74907        41.74502       -20.80403        55.77415
    y2            -49.10609        -9.40081        98.87199        22.56416
    y3             68.29674      -144.83173       -27.35953        15.51142
    y4            121.25932       271.80496        85.85156      -130.11599


                               Parameter Xi Estimates

    Variable              1               2               3               4

    y1             -0.00842        -0.00052        -0.00208        -0.00250
    y2              0.00141         0.00213        -0.00736        -0.00058
    y3             -0.00445         0.00541        -0.00150         0.00310
    y4             -0.00211        -0.00064        -0.00130         0.00197
```

The Johansen cointegration rank test shows whether the series is integrated order either 1 or 2, as shown in Output 30.1.5.

Output 30.1.6. Parameter Estimates

```
                           The VARMAX Procedure

              Type of Model                                VECM(2)
              Estimation Method    Maximum Likelihood Estimation
              Cointegrated Rank                                  1


                          Long-Run Parameter Beta
                           Estimates When RANK=1

                          Variable               1

                          y1               1.00000
                          y2              -0.46458
                          y3              14.51619
                          y4              -9.35520


                           Adjustment Coefficient
                              Alpha Estimates
                                When RANK=1

                          Variable               1

                          y1              -0.01396
                          y2              -0.02811
                          y3              -0.00215
                          y4               0.00510
```

Output 30.1.6 shows that the VECM(2) fits the data. The ECM= option produces the estimates of the long-run parameter, β, and the adjustment coefficient, α.

Output 30.1.7. Parameter Estimates Continued

```
                              The VARMAX Procedure

                               Constant Estimates

                            Variable      Constant

                            y1             0.04076
                            y2             0.08595
                            y3             0.00518
                            y4            -0.01438


                        Parameter Alpha * Beta' Estimates

   Variable              y1               y2               y3               y4

   y1              -0.01396          0.00648         -0.20263          0.13059
   y2              -0.02811          0.01306         -0.40799          0.26294
   y3              -0.00215          0.00100         -0.03121          0.02011
   y4               0.00510         -0.00237          0.07407         -0.04774


                        AR Coefficients of Differenced Lag

DIF Lag   Variable              y1              y2              y3              y4

      1   y1               0.34603         0.09131        -0.35351        -0.96895
          y2               0.09936         0.03791         0.23900         0.28661
          y3               0.18118         0.07859         0.02234         0.40508
          y4               0.03222         0.04961        -0.03292         0.18568
```

Output 30.1.7 shows the parameter estimates in terms of the constant, the one lagged coefficient ($\mathbf{y}_{t-1}$), and the one differenced lagged coefficient ($\Delta\mathbf{y}_{t-1}$).

Output 30.1.8. Parameter Estimates Continued

```
                              The VARMAX Procedure

                           Model Parameter Estimates

                                            Standard
Equation Parameter      Estimate           Error t Value Pr > |t| Variable

D_y1     CONST1          0.04076         0.01418    2.87   0.0048 1
         AR1_1_1        -0.01396         0.00495                  y1(t-1)
         AR1_1_2         0.00648         0.00230                  y2(t-1)
         AR1_1_3        -0.20263         0.07191                  y3(t-1)
         AR1_1_4         0.13059         0.04634                  y4(t-1)
         AR2_1_1         0.34603         0.06414    5.39   0.0001 D_y1(t-1)
         AR2_1_2         0.09131         0.07334    1.25   0.2154 D_y2(t-1)
         AR2_1_3        -0.35351         0.11024   -3.21   0.0017 D_y3(t-1)
         AR2_1_4        -0.96895         0.20737   -4.67   0.0001 D_y4(t-1)
D_y2     CONST2          0.08595         0.01679    5.12   0.0001 1
         AR1_2_1        -0.02811         0.00586                  y1(t-1)
         AR1_2_2         0.01306         0.00272                  y2(t-1)
         AR1_2_3        -0.40799         0.08514                  y3(t-1)
         AR1_2_4         0.26294         0.05487                  y4(t-1)
         AR2_2_1         0.09936         0.07594    1.31   0.1932 D_y1(t-1)
         AR2_2_2         0.03791         0.08683    0.44   0.6632 D_y2(t-1)
         AR2_2_3         0.23900         0.13052    1.83   0.0695 D_y3(t-1)
         AR2_2_4         0.28661         0.24552    1.17   0.2453 D_y4(t-1)
D_y3     CONST3          0.00518         0.01608    0.32   0.7476 1
         AR1_3_1        -0.00215         0.00562                  y1(t-1)
         AR1_3_2         0.00100         0.00261                  y2(t-1)
         AR1_3_3        -0.03121         0.08151                  y3(t-1)
         AR1_3_4         0.02011         0.05253                  y4(t-1)
         AR2_3_1         0.18118         0.07271    2.49   0.0140 D_y1(t-1)
         AR2_3_2         0.07859         0.08313    0.95   0.3463 D_y2(t-1)
         AR2_3_3         0.02234         0.12496    0.18   0.8584 D_y3(t-1)
         AR2_3_4         0.40508         0.23506    1.72   0.0873 D_y4(t-1)
D_y4     CONST4         -0.01438         0.00803   -1.79   0.0758 1
         AR1_4_1         0.00510         0.00281                  y1(t-1)
         AR1_4_2        -0.00237         0.00130                  y2(t-1)
         AR1_4_3         0.07407         0.04072                  y3(t-1)
         AR1_4_4        -0.04774         0.02624                  y4(t-1)
         AR2_4_1         0.03222         0.03632    0.89   0.3768 D_y1(t-1)
         AR2_4_2         0.04961         0.04153    1.19   0.2345 D_y2(t-1)
         AR2_4_3        -0.03292         0.06243   -0.53   0.5990 D_y3(t-1)
         AR2_4_4         0.18568         0.11744    1.58   0.1164 D_y4(t-1)
```

Output 30.1.8 shows the parameter estimates and their significance.

Output 30.1.9. Diagnostic Checks

```
                              The VARMAX Procedure

                           Covariances of Innovations

     Variable               y1               y2               y3               y4

     y1                0.00005          0.00001         -0.00001         -0.00000
     y2                0.00001          0.00007          0.00002          0.00001
     y3               -0.00001          0.00002          0.00007          0.00002
     y4               -0.00000          0.00001          0.00002          0.00002


                                  Information
                                    Criteria

                                AICC    -40.6284
                                HQC     -40.4343
                                AIC     -40.6452
                                SBC     -40.1262
                                FPEC    2.23E-18


           Schematic Representation of Cross Correlations of Residuals
        Variable/
        Lag           0       1       2       3       4       5       6

        y1            ++..    ....    ++..    ....    +...    ..--    ....
        y2            ++++    ....    ....    ....    ....    ....    ....
        y3            .+++    ....    +.-.    ..++    -...    ....    ....
        y4            .+++    ....    ....    ..+.    ....    ....    ....

             + is > 2*std error,  - is < -2*std error,  . is between


                            Portmanteau Test for Cross
                             Correlations of Residuals
                  Up To
                  Lag            DF    Chi-Square    Pr > ChiSq

                        3        16         53.90        <.0001
                        4        32         74.03        <.0001
                        5        48        103.08        <.0001
                        6        64        116.94        <.0001
```

Output 30.1.9 shows the innovation covariance matrix estimates, the various information criteria results, and the tests for white noise residuals.

Output 30.1.10. Diagnostic Checks Continued

```
                                The VARMAX Procedure

                       Univariate Model ANOVA Diagnostics

                                          Standard
           Variable       R-Square        Deviation    F Value    Pr > F

           y1               0.6754          0.00712      32.51    <.0001
           y2               0.3070          0.00843       6.92    <.0001
           y3               0.1328          0.00807       2.39    0.0196
           y4               0.0831          0.00403       1.42    0.1963


                    Univariate Model White Noise Diagnostics

                    Durbin           Normality                  ARCH
    Variable        Watson    Chi-Square    Pr > ChiSq    F Value    Pr > F

    y1             2.13418          7.19        0.0275       1.62    0.2053
    y2             2.04003          1.20        0.5483       1.23    0.2697
    y3             1.86892        253.76        <.0001       1.78    0.1847
    y4             1.98440        105.21        <.0001      21.01    <.0001


                         Univariate Model AR Diagnostics

                  AR1               AR2               AR3               AR4
   Variable  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F

   y1           0.68  0.4126     2.98  0.0542     2.01  0.1154     2.48  0.0473
   y2           0.05  0.8185     0.12  0.8842     0.41  0.7453     0.30  0.8762
   y3           0.56  0.4547     2.86  0.0610     4.83  0.0032     3.71  0.0069
   y4           0.01  0.9340     0.16  0.8559     1.21  0.3103     0.95  0.4358
```

Output 30.1.10 describes how well each univariate equation fits the data.

Output 30.1.11. Infinite Order AR Representation

```
                                The VARMAX Procedure

                         Infinite Order AR Representation

      Lag   Variable           y1            y2            y3            y4

        1   y1            1.33208       0.09780      -0.55614      -0.83836
            y2            0.07125       1.05096      -0.16899       0.54955
            y3            0.17903       0.07959       0.99113       0.42520
            y4            0.03732       0.04724       0.04116       1.13795
        2   y1           -0.34603      -0.09131       0.35351       0.96895
            y2           -0.09936      -0.03791      -0.23900      -0.28661
            y3           -0.18118      -0.07859      -0.02234      -0.40508
            y4           -0.03222      -0.04961       0.03292      -0.18568
        3   y1            0.00000       0.00000       0.00000       0.00000
            y2            0.00000       0.00000       0.00000       0.00000
            y3            0.00000       0.00000       0.00000       0.00000
            y4            0.00000       0.00000       0.00000       0.00000
```

The PRINT=(IARR) option provides the VAR(2) representation.

Output 30.1.12. Weak Exogeneity Test

```
                              The VARMAX Procedure

                   Testing Weak Exogeneity of Each Variables

                 Variable        DF    Chi-Square    Pr > ChiSq

                 y1               1          6.55        0.0105
                 y2               1         12.54        0.0004
                 y3               1          0.09        0.7695
                 y4               1          1.81        0.1786
```

Output 30.1.12 shows whether each variable is the weak exogeneity of other variables. The variable $y1$ is not the weak exogeneity of other variables, $y2$, $y3$, and $y4$; the variable $y2$ is not the weak exogeneity of other variables, $y1$, $y3$, and $y4$.

Example 30.2. Analysis of German Economic Variables

This example considers a three-dimensional VAR(2) model. The model contains the logarithms of a quarterly, seasonally adjusted West German fixed investment, disposable income, and consumption expenditures. The data used are in Lütkepohl (1993, Table E.1).

```
title 'Analysis of German Economic Variables';
data west;
   date = intnx( 'qtr', '01jan60'd, _n_-1 );
   format date yyq. ;
   input y1 y2 y3 @@;
   y1 = log(y1);
   y2 = log(y2);
   y3 = log(y3);
   label y1 = 'logarithm of investment'
         y2 = 'logarithm of income'
         y3 = 'logarithm of consumption';

   datalines;
   ... data lines omitted ...
   ;

data use;
   set west;
   where  date < '01jan79'd;
   keep date y1 y2 y3;

proc varmax data=use;
   id date interval=qtr align=E;
   model y1-y3 / p=2 dify=(1)
                 print=(decompose(6) impulse=(stderr))
```

```
                     printform=both lagmax=3;
      causal group1=(y1) group2=(y2 y3);
      output lead=5;
   run;
```

First the data is fitted to the differenced VAR(2) model as follows.

$$\Delta\mathbf{y}_t = \begin{pmatrix} -0.01672 \\ 0.01577 \\ 0.01293 \end{pmatrix} + \begin{pmatrix} -0.31963 & 0.14599 & 0.96122 \\ 0.04393 & -0.15273 & 0.28850 \\ -0.00242 & 0.22481 & -0.26397 \end{pmatrix} \Delta\mathbf{y}_{t-1} + \begin{pmatrix} -0.16055 & 0.11460 & 0.93439 \\ 0.05003 & 0.01917 & -0.01020 \\ 0.03388 & 0.35491 & -0.02223 \end{pmatrix} \Delta\mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

The parameter estimates AR1_1_2, AR1_1_3, AR2_1_2, and AR2_1_3 are insignificant, and the VARX model is fitted in the next step.

The detailed output is shown in Output 30.2.1 through Output 30.2.8.

Output 30.2.1. Descriptive Statistics

```
                                 The VARMAX Procedure

                  Number of Observations                        75
                  Number of Pairwise Missing                     0
                  Observation(s) eliminated by differencing      1


                              Simple Summary Statistics

                                                   Standard
Variable Type                N          Mean      Deviation            Min            Max

y1       Dependent          75       0.01811        0.04680       -0.14018        0.19358
y2       Dependent          75       0.02071        0.01208       -0.02888        0.05023
y3       Dependent          75       0.01987        0.01040       -0.01300        0.04483

                              Simple Summary Statistics

                  Variable Difference     Label

                  y1                 1     logarithm of investment
                  y2                 1     logarithm of income
                  y3                 1     logarithm of consumption
```

Output 30.2.1 shows the descriptive statistics.

Output 30.2.2. Parameter Estimates

```
                               The VARMAX Procedure

                  Type of Model                              VAR(2)
                  Estimation Method    Least Squares Estimation


                                Constant Estimates

                             Variable      Constant

                             y1             -0.01672
                             y2              0.01577
                             y3              0.01293


                             AR Coefficient Estimates

           Lag    Variable               y1              y2              y3

             1    y1               -0.31963         0.14599         0.96122
                  y2                0.04393        -0.15273         0.28850
                  y3               -0.00242         0.22481        -0.26397
             2    y1               -0.16055         0.11460         0.93439
                  y2                0.05003         0.01917        -0.01020
                  y3                0.03388         0.35491        -0.02223
```

Output 30.2.2 shows that the VAR(2) model fits the data.

Output 30.2.3. Parameter Estimates Continued

```
                              The VARMAX Procedure

                            Schematic Representation
                             of Parameter Estimates

                          Variable/
                          Lag          C    AR1    AR2

                          y1           .    -..    ...
                          y2           +    ...    ...
                          y3           +    .+.    .+.

                             + is > 2*std error,  -
                             is < -2*std error,  .
                             is between,  * is N/A


                           Model Parameter Estimates

                                             Standard
Equation Parameter      Estimate        Error t Value Pr > |t| Variable

y1       CONST1         -0.01672      0.01723   -0.97   0.3352 1
         AR1_1_1        -0.31963      0.12546   -2.55   0.0132 y1(t-1)
         AR1_1_2         0.14599      0.54567    0.27   0.7899 y2(t-1)
         AR1_1_3         0.96122      0.66431    1.45   0.1526 y3(t-1)
         AR2_1_1        -0.16055      0.12491   -1.29   0.2032 y1(t-2)
         AR2_1_2         0.11460      0.53457    0.21   0.8309 y2(t-2)
         AR2_1_3         0.93439      0.66510    1.40   0.1647 y3(t-2)
y2       CONST2          0.01577      0.00437    3.60   0.0006 1
         AR1_2_1         0.04393      0.03186    1.38   0.1726 y1(t-1)
         AR1_2_2        -0.15273      0.13857   -1.10   0.2744 y2(t-1)
         AR1_2_3         0.28850      0.16870    1.71   0.0919 y3(t-1)
         AR2_2_1         0.05003      0.03172    1.58   0.1195 y1(t-2)
         AR2_2_2         0.01917      0.13575    0.14   0.8882 y2(t-2)
         AR2_2_3        -0.01020      0.16890   -0.06   0.9520 y3(t-2)
y3       CONST3          0.01293      0.00353    3.67   0.0005 1
         AR1_3_1        -0.00242      0.02568   -0.09   0.9251 y1(t-1)
         AR1_3_2         0.22481      0.11168    2.01   0.0482 y2(t-1)
         AR1_3_3        -0.26397      0.13596   -1.94   0.0565 y3(t-1)
         AR2_3_1         0.03388      0.02556    1.33   0.1896 y1(t-2)
         AR2_3_2         0.35491      0.10941    3.24   0.0019 y2(t-2)
         AR2_3_3        -0.02223      0.13612   -0.16   0.8708 y3(t-2)
```

Output 30.2.3 shows the parameter estimates and their significance.

Output 30.2.4. Diagnostic Checks

```
                          The VARMAX Procedure

                       Covariances of Innovations

           Variable              y1              y2              y3

           y1               0.00213         0.00007         0.00012
           y2               0.00007         0.00014         0.00006
           y3               0.00012         0.00006         0.00009


                              Information
                                Criteria

                             AICC    -24.4884
                             HQC     -24.2869
                             AIC     -24.5494
                             SBC     -23.8905
                             FPEC    2.18E-11


                    Cross Correlations of Residuals

         Lag    Variable              y1              y2              y3

           0    y1               1.00000         0.13242         0.28275
                y2               0.13242         1.00000         0.55526
                y3               0.28275         0.55526         1.00000
           1    y1               0.01461        -0.00666        -0.02394
                y2              -0.01125        -0.00167        -0.04515
                y3              -0.00993        -0.06780        -0.09593
           2    y1               0.07253        -0.00226        -0.01621
                y2              -0.08096        -0.01066        -0.02047
                y3              -0.02660        -0.01392        -0.02263
           3    y1               0.09915         0.04484         0.05243
                y2              -0.00289         0.14059         0.25984
                y3              -0.03364         0.05374         0.05644


                   Schematic Representation of Cross
                       Correlations of Residuals
                 Variable/
                 Lag           0      1      2      3

                 y1           +.+    ...    ...    ...
                 y2           .++    ...    ...    ..+
                 y3           +++    ...    ...    ...

                      + is > 2*std error,  - is <
                      -2*std error,  . is between


                      Portmanteau Test for Cross
                       Correlations of Residuals
             Up To
             Lag            DF    Chi-Square    Pr > ChiSq

                   3         9          9.69        0.3766
```

Output 30.2.4 shows the innovation covariance matrix estimates, the various information criteria results, and the tests for white noise residuals.

Output 30.2.5. Diagnostic Checks Continued

```
                              The VARMAX Procedure

                    Univariate Model ANOVA Diagnostics

                                         Standard
       Variable       R-Square          Deviation    F Value    Pr > F

       y1               0.1286            0.04615       1.62    0.1547
       y2               0.1142            0.01172       1.42    0.2210
       y3               0.2513            0.00944       3.69    0.0032


                 Univariate Model White Noise Diagnostics

                 Durbin            Normality                  ARCH
Variable         Watson    Chi-Square     Pr > ChiSq    F Value     Pr > F

y1              1.96269         10.22         0.0060      12.39     0.0008
y2              1.98145         11.98         0.0025       0.38     0.5386
y3              2.14583         34.25         <.0001       0.10     0.7480


                      Univariate Model AR Diagnostics

                AR1               AR2               AR3               AR4
Variable  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F

y1           0.01  0.9029     0.19  0.8291     0.39  0.7624     1.39  0.2481
y2           0.00  0.9883     0.00  0.9961     0.46  0.7097     0.34  0.8486
y3           0.68  0.4129     0.38  0.6861     0.30  0.8245     0.21  0.9320
```

Output 30.2.5 describes how well each univariate equation fits the data.

Output 30.2.6. Impulse Response Function

```
                              The VARMAX Procedure

                      Simple Impulse Response by Variable

      Variable    Lag              y1             y2             y3

      y1          1          -0.31963        0.14599        0.96122
                  STD         0.12546        0.54567        0.66431
                  2          -0.05430        0.26174        0.41555
                  STD         0.12919        0.54728        0.66311
                  3           0.11904        0.35283       -0.40789
                  STD         0.08362        0.38489        0.47867
      y2          1           0.04393       -0.15273        0.28850
                  STD         0.03186        0.13857        0.16870
                  2           0.02858        0.11377       -0.08820
                  STD         0.03184        0.13425        0.16250
                  3          -0.00884        0.07147        0.11977
                  STD         0.01583        0.07914        0.09462
      y3          1          -0.00242        0.22481       -0.26397
                  STD         0.02568        0.11168        0.13596
                  2           0.04517        0.26088        0.10998
                  STD         0.02563        0.10820        0.13101
                  3          -0.00055       -0.09818        0.09096
                  STD         0.01646        0.07823        0.10280
```

Output 30.2.6 is the output in a matrix format associated with the PRINT=(IMPULSE=) option for the impulse response function and standard errors.

Output 30.2.7. Proportions of Prediction Error Covariance Decomposition

```
                              The VARMAX Procedure

               Proportions of Prediction Error Covariances by Variable

      Variable       Lead              y1             y2             y3

      y1                1         1.00000        0.00000        0.00000
                        2         0.95996        0.01751        0.02253
                        3         0.94565        0.02802        0.02633
                        4         0.94079        0.02936        0.02985
                        5         0.93846        0.03018        0.03136
                        6         0.93831        0.03025        0.03145
      y2                1         0.01754        0.98246        0.00000
                        2         0.06025        0.90747        0.03228
                        3         0.06959        0.89576        0.03465
                        4         0.06831        0.89232        0.03937
                        5         0.06850        0.89212        0.03938
                        6         0.06924        0.89141        0.03935
      y3                1         0.07995        0.27292        0.64713
                        2         0.07725        0.27385        0.64890
                        3         0.12973        0.33364        0.53663
                        4         0.12870        0.33499        0.53631
                        5         0.12859        0.33924        0.53217
                        6         0.12852        0.33963        0.53185
```

The proportions of decomposition of the prediction error covariances of three variables are given in Output 30.2.7.

Output 30.2.8. Forecasts

```
                                 The VARMAX Procedure

                                      Forecasts

                                                   Standard
Variable      Obs   Time      Forecast        Error    95% Confidence Limits

y1             77 1979:1       6.54027      0.04615      6.44982      6.63072
               78 1979:2       6.55105      0.05825      6.43688      6.66522
               79 1979:3       6.57217      0.06883      6.43725      6.70708
               80 1979:4       6.58452      0.08021      6.42732      6.74173
               81 1980:1       6.60193      0.09117      6.42324      6.78063
y2             77 1979:1       7.68473      0.01172      7.66176      7.70770
               78 1979:2       7.70508      0.01691      7.67193      7.73822
               79 1979:3       7.72206      0.02156      7.67980      7.76431
               80 1979:4       7.74266      0.02615      7.69140      7.79392
               81 1980:1       7.76240      0.03005      7.70350      7.82130
y3             77 1979:1       7.54024      0.00944      7.52172      7.55875
               78 1979:2       7.55489      0.01282      7.52977      7.58001
               79 1979:3       7.57472      0.01808      7.53928      7.61015
               80 1979:4       7.59344      0.02205      7.55022      7.63666
               81 1980:1       7.61232      0.02578      7.56179      7.66286
```

The table in Output 30.2.8 gives forecasts and their prediction error covariances.

Output 30.2.9. Granger-Causality Tests

```
                                 The VARMAX Procedure

                              Granger-Causality Wald Test

                        Test        DF    Chi-Square    Pr > ChiSq

                           1         4          6.37        0.1734


                           Test 1:  Group 1 Variables:  y1
                                    Group 2 Variables:  y2 y3
```

Output 30.2.9 shows that you cannot reject Granger-noncausality from $(y2, y3)$ to $y1$ using the 0.05 significance level.

The following SAS statements show that the variable $y1$ is the exogenous variable, and fit the VARX(2,1) model to the data.

```
   proc varmax data=use;
      id date interval=qtr align=E;
      model y2 y3 = y1 / p=2 dify=(1) difx=(1) xlag=1
                         lagmax=3;
   run;
```

The fitted VARX(2,1) model is written as

$$\begin{pmatrix} \Delta y_{2t} \\ \Delta y_{3t} \end{pmatrix} = \begin{pmatrix} 0.01542 \\ 0.01319 \end{pmatrix} + \begin{pmatrix} 0.02520 \\ 0.05130 \end{pmatrix} \Delta y_{1t} + \begin{pmatrix} 0.03870 \\ 0.00363 \end{pmatrix} \Delta y_{1,t-1}$$
$$+ \begin{pmatrix} -0.12258 & 0.25811 \\ 0.24367 & -0.31809 \end{pmatrix} \begin{pmatrix} \Delta y_{2,t-1} \\ \Delta y_{3,t-1} \end{pmatrix}$$
$$+ \begin{pmatrix} 0.01651 & 0.03498 \\ 0.34921 & -0.01664 \end{pmatrix} \begin{pmatrix} \Delta y_{2,t-2} \\ \Delta y_{3,t-2} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{pmatrix}$$

The detailed output is shown in Output 30.2.10 through Output 30.2.13.

Output 30.2.10. Parameter Estimates

```
                          The VARMAX Procedure

                 Type of Model                         VARX(2,1)
                 Estimation Method    Least Squares Estimation


                           Constant Estimates

                        Variable      Constant

                        y2             0.01542
                        y3             0.01319


                        Coefficient Estimates of
                         Independent Variables

                      Lag    Variable              y1

                        0    y2               0.02520
                             y3               0.05130
                        1    y2               0.03870
                             y3               0.00363


                        AR Coefficient Estimates

              Lag    Variable              y2              y3

                1    y2              -0.12258         0.25811
                     y3               0.24367        -0.31809
                2    y2               0.01651         0.03498
                     y3               0.34921        -0.01664
```

Output 30.2.10 shows the parameter estimates in terms of the constant, the current and the one lagged coefficient of the exogenous variable, and the two lagged coefficients of the dependent variables.

Output 30.2.11. Parameter Estimates Continued

```
                              The VARMAX Procedure

                           Model Parameter Estimates

                                           Standard
Equation Parameter       Estimate         Error t Value Pr > |t| Variable

y2       CONST1           0.01542       0.00443    3.48   0.0009 1
         XL0_1_1          0.02520       0.03130    0.81   0.4237 y1(t)
         XL1_1_1          0.03870       0.03252    1.19   0.2383 y1(t-1)
         AR1_1_1         -0.12258       0.13903   -0.88   0.3811 y2(t-1)
         AR1_1_2          0.25811       0.17370    1.49   0.1421 y3(t-1)
         AR2_1_1          0.01651       0.13766    0.12   0.9049 y2(t-2)
         AR2_1_2          0.03498       0.16783    0.21   0.8356 y3(t-2)
y3       CONST2           0.01319       0.00346    3.81   0.0003 1
         XL0_2_1          0.05130       0.02441    2.10   0.0394 y1(t)
         XL1_2_1          0.00363       0.02536    0.14   0.8868 y1(t-1)
         AR1_2_1          0.24367       0.10842    2.25   0.0280 y2(t-1)
         AR1_2_2         -0.31809       0.13546   -2.35   0.0219 y3(t-1)
         AR2_2_1          0.34921       0.10736    3.25   0.0018 y2(t-2)
         AR2_2_2         -0.01664       0.13088   -0.13   0.8992 y3(t-2)
```

Output 30.2.11 shows the parameter estimates and their significance.

Output 30.2.12. Diagnostic Checks

```
                             The VARMAX Procedure

                          Covariances of Innovations

                   Variable              y2              y3

                   y2               0.00014         0.00006
                   y3               0.00006         0.00009


                                 Information
                                   Criteria

                             AICC    -18.3902
                             HQC     -18.2558
                             AIC     -18.4309
                             SBC     -17.9916
                             FPEC     9.91E-9


                         Cross Correlations of Residuals

                 Lag    Variable              y2              y3

                   0    y2               1.00000         0.56462
                        y3               0.56462         1.00000
                   1    y2              -0.02312        -0.05927
                        y3              -0.07056        -0.09145
                   2    y2              -0.02849        -0.05262
                        y3              -0.05804        -0.08567
                   3    y2               0.16071         0.29588
                        y3               0.10882         0.13002


                        Schematic Representation of Cross
                            Correlations of Residuals
                        Variable/
                        Lag          0     1     2     3

                        y2           ++    ..    ..    .+
                        y3           ++    ..    ..    ..

                           + is > 2*std error,  - is <
                           -2*std error,  . is between


                           Portmanteau Test for Cross
                            Correlations of Residuals
                Up To
                Lag            DF    Chi-Square    Pr > ChiSq

                      3         4          8.38        0.0787
```

Output 30.2.12 shows the innovation covariance matrix estimates, the various information criteria results, and the tests for white noise residuals.

Output 30.2.13. Diagnostic Checks Continued

```
                         The VARMAX Procedure

                  Univariate Model ANOVA Diagnostics

                                       Standard
       Variable       R-Square        Deviation     F Value     Pr > F

       y2               0.0897          0.01188        1.08     0.3809
       y3               0.2796          0.00926        4.27     0.0011


                Univariate Model White Noise Diagnostics

                    Durbin           Normality                 ARCH
Variable            Watson     Chi-Square     Pr > ChiSq    F Value     Pr > F

y2                 2.02413          14.54         0.0007       0.49     0.4842
y3                 2.13414          32.27         <.0001       0.08     0.7782


                     Univariate Model AR Diagnostics

                AR1               AR2               AR3               AR4
Variable  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F  F Value  Pr > F

y2           0.04  0.8448     0.04  0.9570     0.62  0.6029     0.42  0.7914
y3           0.62  0.4343     0.62  0.5383     0.72  0.5452     0.36  0.8379
```

Output 30.2.13 describes how well each univariate equation fits the data.

Example 30.3. Numerous Examples

The following are examples of syntax for model fitting:

```
/* Data ’a’ Generated Process */
  proc iml;
     sig = {1.0  0.5, 0.5  1.25};
     phi = {1.2 -0.5, 0.6  0.3};
     call varmasim(y,phi) sigma = sig n = 100 seed = 46859;
     cn = {’y1’ ’y2’};
     create a from y[colname=cn];
     append from y;
  quit;

/* when the series has a linear trend */
  proc varmax data=a;
     model y1 y2 / p=1 trend=linear;
  run;

/* Fit subset of AR order 1 and 3 */
  proc varmax data=a;
     model y1 y2 / p=(1,3);
  run;
```

```
/* Check if the series is nonstationary */
  proc varmax data=a;
     model y1 y2 / p=1 dftest print=(roots);
  run;

/* Fit VAR(1) in differencing */
  proc varmax data=a;
     model y1 y2 / p=1 print=(roots) dify=(1);
  run;

/* Fit VAR(1) in seasonal differencing */
  proc varmax data=a;
     model y1 y2 / p=1 dify=(4) lagmax=5;
  run;

/* Fit VAR(1) in both regular and seasonal differencing */
  proc varmax data=a;
     model y1 y2 / p=1 dify=(1,4) lagmax=5;
  run;

/* Fit VAR(1) in different differencing */
  proc varmax data=a;
     model y1 y2 / p=1 dif=(y1(1,4) y2(1)) lagmax=5;
  run;

/* Options related prediction */
  proc varmax data=a;
     model y1 y2 / p=1 lagmax=3
                   print=(impulse covpe(5) decompose(5));
  run;

/* Options related tentative order selection */
  proc varmax data=a;
     model y1 y2 / p=1 lagmax=5 minic
                   print=(parcoef pcancorr pcorr);
  run;

/* Automatic selection of the AR order */
  proc varmax data=a;
     model y1 y2 / minic=(type=aic p=5);
  run;

/* Compare results of LS and Yule-Walker Estimator */
  proc varmax data=a;
     model y1 y2 / p=1 print=(yw);
  run;

/* BVAR(1) of the nonstationary series y1 and y2 */
  proc varmax data=a;
     model y1 y2 / p=1
        prior=(lambda=1 theta=0.2 ivar nrep=200);
  run;

/* BVAR(1) of the nonstationary series y1 */
```

```
  proc varmax data=a;
     model y1 y2 / p=1
        prior=(lambda=0.1 theta=0.15 ivar=(y1) seed=12345);
  run;

/* Data ’b’ Generated Process */
  proc iml;
     sig = { 0.5  0.14 -0.08 -0.03,  0.14 0.71 0.16 0.1,
            -0.08 0.16  0.65  0.23, -0.03 0.1  0.23 0.16};
     sig = sig * 0.0001;
     phi = {1.2 -0.5 0.  0.1,  0.6 0.3 -0.2  0.5,
            0.4  0. -0.2 0.1, -1.0 0.2  0.7 -0.2};
     call varmasim(y,phi) sigma = sig n = 100 seed = 32567;
     cn = {’y1’ ’y2’ ’y3’ ’y4’};
     create b from y[colname=cn];
     append from y;
  quit;

/* Cointegration Rank Test using Trace statistics */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest;
  run;

/* Cointegration Rank Test using Max statistics */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(type=max));
  run;

/* Common Trends Test using Filter(Differencing) statistics */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest=(sw);
  run;

/* Common Trends Test using Filter(Residual) statistics */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=filtres lag=1));
  run;

/* Common Trends Test using Kernel statistics */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=kernel lag=1));
  run;

/* Cointegration Rank Test for I(2) */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(iorder=2));
  run;

/* Fit VECM(2) with rank=3 */
  proc varmax data=b;
     model y1-y4 / p=2 lagmax=4 print=(roots iarr)
                  ecm=(rank=3 normalize=y1);
  run;
```

```
/* Weak Exogenous Testing for each variable */
  proc varmax data=b outstat=bbb;
     model y1-y4 / p=2 lagmax=4
                   ecm=(rank=3 normalize=y1);
     cointeg rank=3 exogeneity;
  run;

/* Hypotheses Testing for long-run and adjustment parameter */
  proc varmax data=b outstat=bbb;
     model y1-y4 / p=2 lagmax=4
                   ecm=(rank=3 normalize=y1);
     cointeg rank=3 normalize=y1
        h=(1 0 0, 0 1 0, -1 0 0, 0 0 1)
        j=(1 0 0, 0 1 0, 0 0 1, 0 0 0);
  run;

/* ordinary regression model */
  proc varmax data=grunfeld;
     model y1 y2 = x1-x3;
  run;

/* Ordinary regression model with subset lagged terms */
  proc varmax data=grunfeld;
     model y1 y2 = x1 / xlag=(1,3);
  run;

/* VARX(1,1) with no current time Exogenous Variables */
  proc varmax data=grunfeld;
     model y1 y2 = x1 / p=1 xlag=1 nocurrentx;
  run;

/* VARX(1,1) with different Exogenous Variables */
  proc varmax data=grunfeld;
     model y1 = x3, y2 = x1 x2 / p=1 xlag=1;
  run;

/* VARX(1,2) in difference with current Exogenous Variables */
  proc varmax data=grunfeld;
     model y1 y2 = x1 / p=1 xlag=2 difx=(1) dify=(1);
  run;
```

Example 30.4. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics.

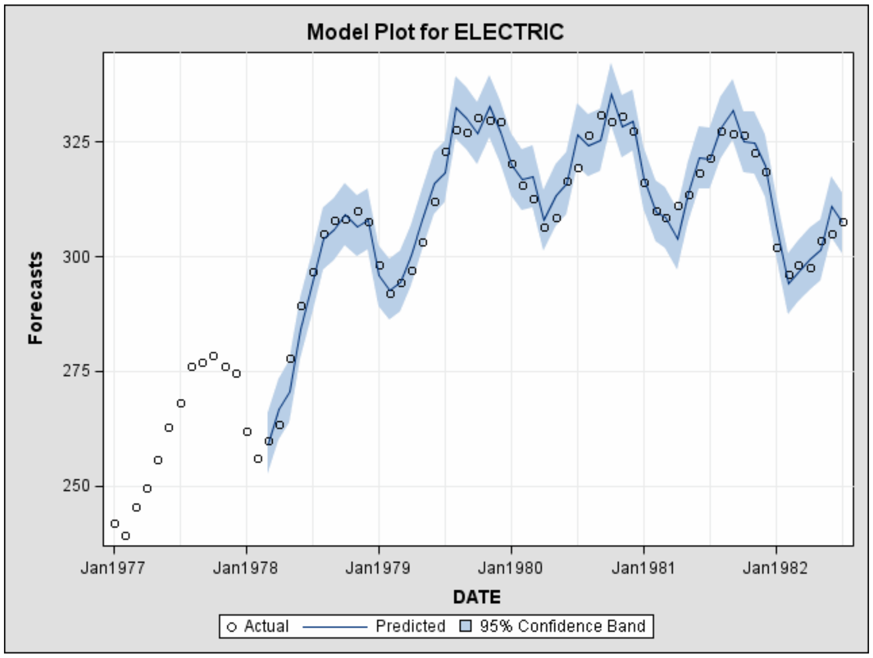
The following statements use the SASHELP.WORKERS data set to study the time series of electrical workers and its interaction with the series of masonry workers. The series and predict plots, the residual plot, and the forecast plot are created in Output 30.4.1 through Output 30.4.3. These are a selection of the plots created by the VARMAX procedure.

The graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter

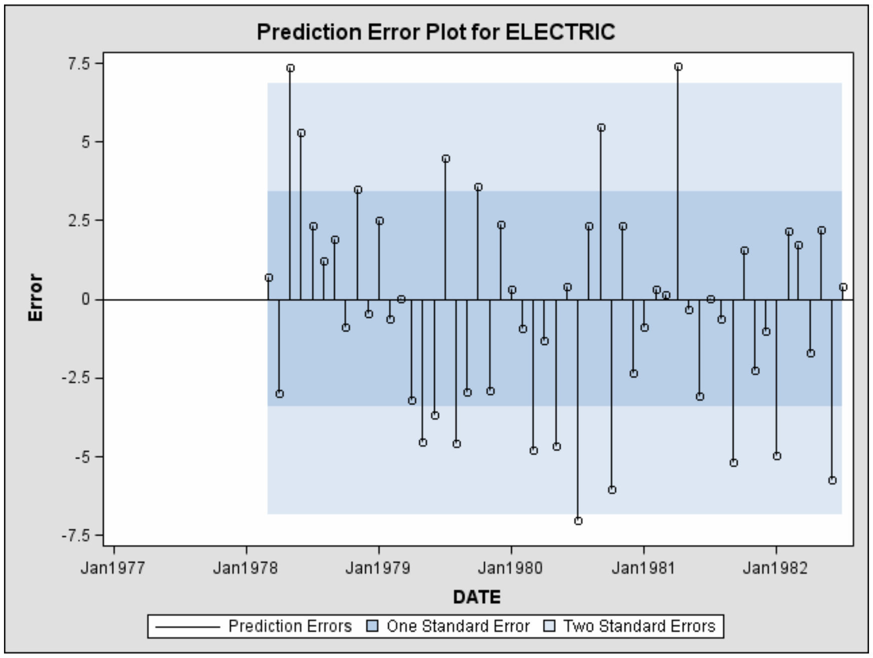
9, “Statistical Graphics Using ODS.” For specific information about the graphics available in the VARMAX procedure, see the “ODS Graphics” section on page 1825.

```
ods html;
ods graphics on;

title "Illustration of ODS Graphics";
proc varmax data=sashelp.workers;
   id date interval=month;
   model electric masonry / dify=(1,12) noint p=1;
   output lead=12;
run;

ods graphics off;
ods html close;
```

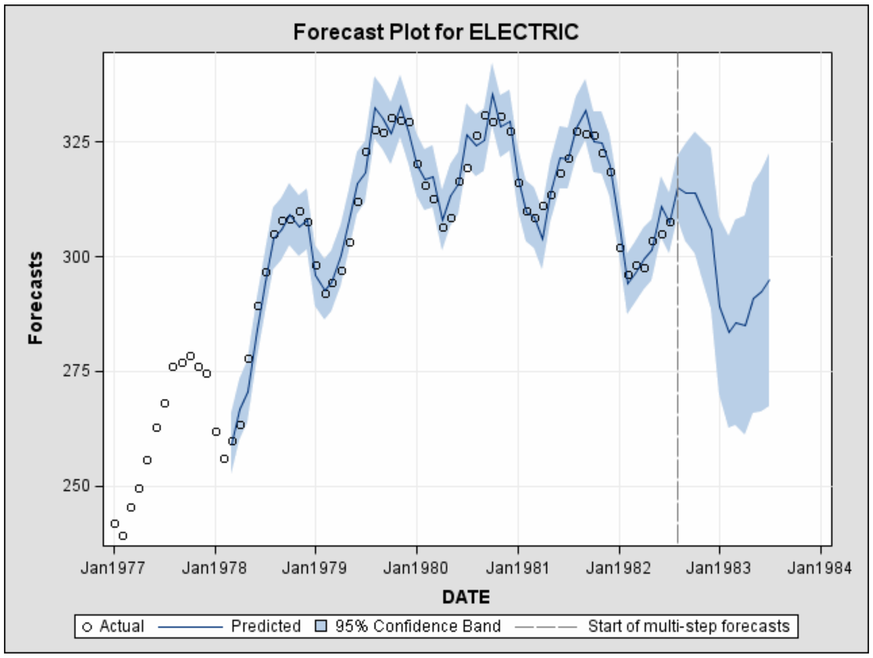
Output 30.4.1. Series and Predicted Series Plots (Experimental)



Output 30.4.2. Residual Plot (Experimental)



Output 30.4.3. Series and Forecast Plots (Experimental)



References

Ahn, S. K. and Reinsel, G. C. (1988), "Nested Reduced-Rank Autoregressive Models for Multiple Time Series," *Journal of the American Statistical Association*, 83, 849–856.

Anderson, T. W. (1951), "Estimating Linear Restrictions on Regression Coefficients for Multivariate Normal Distributions," *Annals of Mathematical Statistics*, 22, 327-351.

Ansley, C. F. and Kohn, R. (1986), "A Note on Reparameterizing a Vector Autoregressive Moving Average Model to Enforce Stationarity," *Journal of Statistical Computation and Simulation*, 24, 99–106.

Ansley, C. F., Kohn, R., and Shively, T. S. (1992), "Computing *p*-values for the Generalized Durbin-Watson and Other Invariant Test Statistics," *Journal of Econometrics*, 54, 277–300.

Bollerslev, T., Engle, R. F., and Wooldridge, J. M. (1988), "A Capital Asset Pricing Model with Time Varying Covariances," *Journal of Political Economy*, 96, 116–131.

Brockwell, P. J. and Davis, R. A. (1991), *Time Series: Theory and Methods*, Second Edition, New York: Springer-Verlag.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984), "Testing for Unit Roots in Seasonal Time Series," *Journal of the American Statistical Association*, 79, 355–367.

Doan, T., Litterman, R., and Sims, C. (1984), "Forecasting and Conditional Projection Using Realistic Prior Distributions," *Econometric Reviews*, 3, 1–144.

Engle, R. F. and Granger, C. W. J. (1987), "Co-integration and Error Correction: Representation, Estimation and Testing," *Econometrica*, 55, 251–276.

Engle, R. F. and Kroner, K. F. (1995), "Multivariate Simultaneous Generalized ARCH," *Econometric Theory*, 11, 122–150.

Engle, R. F. and Yoo, B. S. (1987), "Forecasting and Testing in Co-Integrated Systems," *Journal of Econometrics*, 35, 143–159.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton: Princeton University Press.

Hannan, E. J. (1970), *Multiple Time Series*, New York: John Wiley & Sons.

Hannan, E. J. and Deistler, M. (1988), *The Statistical Theory of Linear Systems*, New York: John Wiley & Sons.

Hosking, J. R. M. (1980), "The Multivariate Portmanteau Statistic," *Journal of the American Statistical Association*, 75, 602–608.

Hurvichm, C. M. and Tsai, C. (1993), "A Corrected Akaike Information Criterion for Vector Autoregressive Model Selection," *Journal of Time Series Analysis*, 14, 271–279.

Johansen, S. (1988), “Statistical Analysis of Cointegration Vectors,” *Journal of Economic Dynamics and Control*, 12, 231–254.

Johansen, S. (1992a), “A Representation of Vector Autoregressive Processes Integrated of Order 2,” *Econometric Theory*, 8, 188–202.

Johansen, S. (1992b), “Testing Weak Exogeneity and the Order of Cointegration in UK Money Demand Data,” *Journal of Policy Modeling*, 14, 313–334.

Johansen, S. (1995a), “A Statistical Analysis of Cointegration for I(2) Variables,” *Econometric Theory*, 11, 25–59.

Johansen, S. (1995b), *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*, New York: Oxford University Press.

Johansen, S. and Juselius, K. (1990), “Maximum Likelihood Estimation and Inference on Cointegration: With Applications to the Demand for Money,” *Oxford Bulletin of Economics and Statistics*, 52, 169–210.

Johansen, S. and Juselius, K. (1992), “Testing Structural Hypotheses in a Multivariate Cointegration Analysis of the PPP and the UIP for UK,” *Journal of Econometrics*, 53, 211–244.

Kim, M. (1996), “A Remark on Algorithm AS 279: Computing p-values for the Generalized Durbin-Watson Statistic and Residual Autocorrelation in Regression,” *Applied Statistics*, 45, 273–274.

Kohn, R., Shively, T. S., and Ansley, C. F. (1993), “Algorithm AS 279: Computing p-values for the Generalized Durbin-Watson Statistic and Residual Autocorrelation in Regression,” *Applied Statistics*, 42, 249–269.

Koreisha, S. and Pukkila, T. (1989), “Fast Linear Estimation Methods for Vector Autoregressive Moving Average Models,” *Journal of Time Series Analysis*, 10, 325-339.

Litterman, R. B. (1986), “Forecasting with Bayesian Vector Autoregressions: Five Years of Experience,” *Journal of Business & Economic Statistics*, 4, 25–38.

Lütkepohl, H. (1993), *Introduction to Multiple Time Series Analysis*, Berlin: Springer-Verlag.

Maddala, G. S. and Kim, I. (1998), *Unit Roots, Cointegration, and Structural Change*, Cambridge: Cambridge University Press.

Osterwald-Lenum, M. (1992), “A Note with Quantiles of the Asymptotic Distribution of the Maximum Likelihood Cointegration Rank Test Statistics,” *Oxford Bulletin of Economics and Statistics*, 54, 461–472.

Pringle, R. M. and Raynor, D. L. (1971), *Generalized Inverse Matrices with Applications to Statistics*, Second Edition, New York: McGraw-Hill Inc.

Reinsel, G. C. (1997), *Elements of Multivariate Time Series Analysis*, Second Edition, New York: Springer-Verlag.

Quinn, B. G. (1980), “Order Determination for a Multivariate Autoregression,” *Journal of the Royal Statistical Society*, B, 42, 182–185.

Spliid, H. (1983), "A Fast Estimation for the Vector Autoregressive Moving Average Models with Exogenous Variables," *Journal of the American Statistical Association*, 78, 843–849.

Stock, J. H. and Watson, M. W. (1988), "Testing for Common Trends," *Journal of the American Statistical Association*, 83, 1097–1107.

Todd, R. M. (1984), "Improving Economic Forecasting with Bayesian Vector Autoregression," *Federal Reserve Bank of Minneapolis Quarterly Review*, 4 (Fall), 18–29.

Chapter 31
The X11 Procedure

Chapter Contents

Chapter 31
The X11 Procedure

Overview

The X11 procedure, an adaptation of the U.S. Bureau of the Census X-11 Seasonal Adjustment program, seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set containing the adjusted time series and intermediate calculations.

The X11 procedure also provides the X-11-ARIMA method developed by Statistics Canada. This method fits an ARIMA model to the original series, then uses the model forecast to extend the original series. This extended series is then seasonally adjusted by the standard X-11 seasonal adjustment method. The extension of the series improves the estimation of the seasonal factors and reduces revisions to the seasonally adjusted series as new data becomes available.

The X11 procedure incorporates Sliding Spans Analysis. This type of analysis provides a diagnostic for determining the suitability of seasonal adjustment for an economic series.

Seasonal adjustment of a series is based on the assumption that seasonal fluctuations can be measured in the original series, O_t, t=1,...,n, and separated from trend cycle, trading-day, and irregular fluctuations. The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated constantly or in an evolving fashion from year to year. The trend cycle component, C_t, includes variation due to the long-term trend, the business cycle, and other long-term cyclical factors. The trading-day component, D_t, is the variation that can be attributed to the composition of the calendar. The irregular component, I_t, is the residual variation. Many economic time series are related in a multiplicative fashion ($O_t = S_t C_t D_t I_t$). A seasonally adjusted time series, $C_t I_t$, consists of only the trend cycle and irregular components.

Getting Started

The most common use of the X11 procedure is to produce a seasonally adjusted series. Eliminating the seasonal component from an economic series facilitates comparison among consecutive months or quarters. A plot of the seasonally adjusted series is often more informative about trends or location in a business cycle than a plot of the unadjusted series.

The following example shows how to use PROC X11 to produce a seasonally adjusted series, C_tI_t from an original series $O_t = S_tC_tD_tI_t$.

In the multiplicative model, the trend cycle component C_t keeps the same scale as the original series O_t, while S_t, D_t, and I_t vary around 1.0. In all printed tables and in the output data set, these latter components are expressed as percentages, and thus will vary around 100.0 (in the additive case, they vary around 0.0).

The naming convention used in PROC X11 for the tables follows the original U.S. Bureau of the Census X-11 Seasonal Adjustment program specification; refer the U.S. Bureau of the Census, 1967, and "Printed Output" later in this chapter. This convention is outlined in Figure 31.1.

The tables corresponding to parts A - C are intermediate calculations. The final estimates of the individual components are found in the D tables: table D10 contains the final seasonal factors, table D12 contains the final trend cycle, and table D13 contains the final irregular series. If you are primarily interested in seasonally adjusting a series without consideration of intermediate calculations or diagnostics, you only need to look at table D11, the final seasonally adjusted series.

Basic Seasonal Adjustment

Suppose you have monthly retail sales data starting in September, 1978, in a SAS data set named SALES. At this point you do not suspect that any calendar effects are present and there are no prior adjustments that need to be made to the data.

In this simplest case, you need only specify the DATE= variable in the MONTHLY statement, which associates a SAS date value to each observation. To see the results of the seasonal adjustment, you must request table D11, the final seasonally adjusted series, in a TABLES statement.

```
data sales;
   input sales @@;
   date = intnx( 'month', '01sep1978'd, _n_-1 );
   format date monyy7.;
   datalines;
run;

proc x11 data=sales;
   monthly date=date;
   var sales;
   tables d11;
run;
```

```
                           The X11 Procedure

                    X-11 Seasonal Adjustment Program
                       U. S. Bureau of the Census
                 Economic Research and Analysis Division
                           November 1, 1968

           The X-11 program is divided into seven major parts.
            Part         Description
             A.  Prior adjustments, if any
             B.  Preliminary estimates of irregular component weights
                     and regression trading day factors
             C.  Final estimates of above
             D.  Final estimates of seasonal, trend-cycle and
                     irregular components
             E.  Analytical tables
             F.  Summary measures
             G.  Charts

                            Series - sales
                  Period covered - 9/1978 to 8/1990
              Type of run: multiplicative seasonal adjustment.
                        No printout.   No charts.
           Sigma limits for graduating extreme values are 1.5 and 2.5
           Irregular values outside of 2.5-sigma limits are excluded
                           from trading day regression
```

```
                              The X11 Procedure

                        Seasonal Adjustment of - sales

                  D11 Final Seasonally Adjusted Series
Year       JAN        FEB        MAR        APR        MAY        JUN

1978         .          .          .          .          .          .
1979   124.935    126.533    125.282    125.650    127.754    129.648
1980   128.734    139.542    143.726    143.854    148.723    144.530
1981   176.329    166.264    167.433    167.509    173.573    175.541
1982   186.747    202.467    192.024    202.761    197.548    206.344
1983   233.109    223.345    218.179    226.389    224.249    227.700
1984   238.261    239.698    246.958    242.349    244.665    247.005
1985   275.766    282.316    294.169    285.034    294.034    296.114
1986   325.471    332.228    330.401    330.282    333.792    331.349
1987   363.592    373.118    368.670    377.650    380.316    376.297
1988   370.966    384.743    386.833    405.209    380.840    389.132
1989   428.276    418.236    429.409    446.467    437.639    440.832
1990   480.631    474.669    486.137    483.140    481.111    499.169
---------------------------------------------------------------------
Avg    277.735    280.263    282.435    286.358    285.354    288.638

                     D11 Final Seasonally Adjusted Series
Year       JUL       AUG       SEP       OCT       NOV       DEC     Total

1978         .         .   123.507   125.776   124.735   129.870   503.887
1979   127.880   129.285   126.562   134.905   133.356   136.117   1547.91
1980   140.120   153.475   159.281   162.128   168.848   165.159   1798.12
1981   179.301   182.254   187.448   197.431   184.341   184.304   2141.73
1982   211.690   213.691   214.204   218.060   228.035   240.347   2513.92
1983   222.045   222.127   222.835   212.227   230.187   232.827   2695.22
1984   251.247   253.805   264.924   266.004   265.366   277.025   3037.31
1985   294.196   309.162   311.539   319.518   318.564   323.921   3604.33
1986   337.095   341.127   346.173   350.183   360.792   362.333   4081.23
1987   379.668   375.607   374.257   372.672   368.135   364.150   4474.13
1988   385.479   377.147   397.404   403.156   413.843   416.142   4710.89
1989   450.103   454.176   460.601   462.029   427.499   485.113   5340.38
1990   485.370   485.103         .         .         .         .   3875.33
--------------------------------------------------------------------------
Avg    288.683   291.413   265.728   268.674   268.642   276.442

               Total:  40324  Mean:  280.03  S.D.:  111.31
```

Figure 31.1. Basic Seasonal Adjustment

You can compare the original series, table B1, and the final seasonally adjusted series, table D11 by plotting them together. These tables are requested and named in the OUTPUT statement.

```
title 'Monthly Retail Sales Data (in $1000)';

proc x11 data=sales noprint;
   monthly date=date;
   var sales;
   output out=out b1=sales d11=adjusted;
run;

symbol1 i=join v='star';
symbol2 i=join v='circle';
legend1 label=none value=('original' 'adjusted');
```

```
proc gplot data=out;
   plot sales    * date = 1
        adjusted * date = 2 / overlay legend=legend1;
run;
```

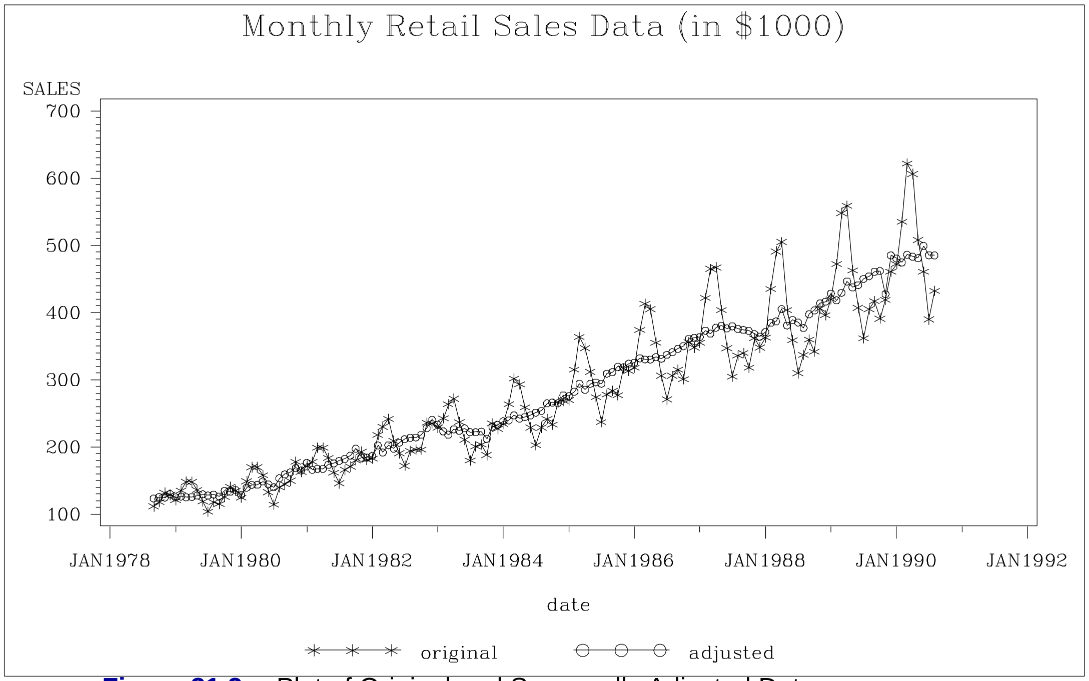


Figure 31.2. Plot of Original and Seasonally Adjusted Data

X-11-ARIMA

An inherent problem with the X-11 method is the revision of the seasonal factor estimates as new data becomes available. The X-11 method uses a set of centered moving averages to estimate the seasonal components. These moving averages apply symmetric weights to all observations except those at the beginning and end of the series, where asymmetric weights have to be applied. These asymmetric weights can cause poor estimates of the seasonal factors, which then can cause large revisions when new data becomes available.

While large revisions to seasonally adjusted values are not common, they can happen. When they do happen, it undermine the credibility of the X-11 seasonal adjustment method.

A method to address this problem was developed at Statistics Canada (Dagum, 1980, 1982b). This method, known as X-11-ARIMA, applies an ARIMA model to the original data (after adjustments, if any) to forecast the series one or more years. This extended series is then seasonally adjusted, allowing symmetric weights to be applied to the end of the original data. This method was tested against a large number of Canadian economic series and was found to greatly reduce the amount of revisions as new data were added.

The X-11-ARIMA method is available in PROC X11 through the use of the ARIMA statement. The ARIMA statement extends the original series either with a user-specified ARIMA model or by an automatic selection process in which the best model from a set of five predefined ARIMA models is used.

The following example illustrates the use of the ARIMA statement. The ARIMA statement does not contain a user-specified model, so the best model is chosen by the automatic selection process. Forecasts from this best model are then used to extend the original series by one year. The partial listing below shows parameter estimates and model diagnostics for the ARIMA model chosen by the automatic selection process.

```
proc x11 data=sales;
   monthly date=date;
   var sales;
   arima;
run;
```

```
                           The X11 Procedure

                     Seasonal Adjustment of - sales

                  Conditional Least Squares Estimation
                                        Approx.
          Parameter      Estimate    Std Error    t Value    Lag

          MU            0.0001728    0.0009596       0.18      0
          MA1,1         0.3739984    0.0893427       4.19      1
          MA1,2         0.0231478    0.0892154       0.26      2
          MA2,1         0.5727914    0.0790835       7.24     12


                  Conditional Least Squares Estimation

                 Variance  Estimate =    0.0014313
                 Std Error Estimate =    0.0378326
                 AIC                =    -482.2412    *
                 SBC                =    -470.7404    *
                 Number of Residuals=          131

                   * Does not include log determinant


           Criteria Summary for Model 2: (0,1,2)(0,1,1)s, Log Transform

                 Box-Ljung Chi-square: 22.03 with 21 df Prob= 0.40
                              (Criteria prob > 0.05)
              Test for over-differencing: sum of MA parameters = 0.57
                                 (must be < 0.90)
               MAPE - Last Three Years:     2.84 (Must be < 15.00 %)
                              - Last Year:            3.04
                              - Next to Last Year:    1.96
                              - Third from Last Year: 3.51
```

Figure 31.3. X-11-ARIMA Model Selection

Table D11 (final seasonally adjusted series) is now constructed using symmetric weights on observations at the end of the actual data. This should result in better estimates of the seasonal factors and, thus, smaller revisions in D11 as more data become available.

Syntax

The X11 procedure uses the following statements:

PROC X11 *options*;
  ARIMA *options*;
  BY *variables*;
  ID *variables*;
  MACURVES *option*;
  MONTHLY *options*;
  OUTPUT *OUT=dataset options*;
  PDWEIGHTS *option*;
  QUARTERLY *options*;
  SSPAN *options* ;
  TABLES *tablenames*;
  VAR *variables*;

Either the MONTHLY or QUARTERLY statement must be specified, depending on the type of time series data you have. The PDWEIGHTS and MACURVES statements can be used only with the MONTHLY statement. The TABLES statement controls the printing of tables, while the OUTPUT statement controls the creation of the OUT= data set.

Functional Summary

The statements and options controlling the X11 procedures are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify input data set	PROC X11	DATA=
write the trading-day regression results to an output data set	PROC X11	OUTTDR=
write the stable seasonality test results to an output data set	PROC X11	OUTSTB=
write table values to an output data set	OUTPUT	OUT=
add extrapolated values to the output data set	PROC X11	OUTEX
add year ahead estimates to the output data set	PROC X11	YRAHEADOUT
write the sliding spans analysis results to an output data set	PROC X11	OUTSPAN=

Description	Statement	Option
Printing Control Options		
suppress all printed output	PROC X11	NOPRINT
suppress all printed ARIMA output	ARIMA	NOPRINT
print all ARIMA output	ARIMA	PRINTALL
print selected tables and charts	TABLES	
print selected groups of tables	MONTHLY	PRINTOUT=
	QUARTERLY	PRINTOUT=
print selected groups of charts	MONTHLY	CHARTS=
	QUARTERLY	CHARTS=
print preliminary tables associated with ARIMA processing	ARIMA	PRINTFP
specify number of decimals for printed tables	MONTHLY	NDEC=
	QUARTERLY	NDEC=
suppress all printed SSPAN output	SSPAN	NOPRINT
print all SSPAN output	SSPAN	PRINTALL
Date Information Options		
specify a SAS Date variable	MONTHLY	DATE=
	QUARTERLY	DATE=
specify the beginning date	MONTHLY	START=
	QUARTERLY	START=
specify the ending date	MONTHLY	END=
	QUARTERLY	END=
specify beginning year for trading-day regression	MONTHLY	TDCOMPUTE=
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the variables to be seasonally adjusted	VAR	
specify identifying variables	ID	
specify the prior monthly factor	MONTHLY	PMFACTOR=
Controlling the table computations		
use additive adjustment	MONTHLY	ADDITIVE
	QUARTERLY	ADDITIVE
specify seasonal factor moving average length	MACURVES	
specify the extreme value limit for trading-day regression	MONTHLY	EXCLUDE=
specify the lower bound for extreme irregulars	MONTHLY	FULLWEIGHT=
	QUARTERLY	FULLWEIGHT=
specify the upper bound for extreme irregulars	MONTHLY	ZEROWEIGHT=
	QUARTERLY	ZEROWEIGHT=

Description	Statement	Option
include the length-of-month in trading-day regression	MONTHLY	LENGTH
specify trading-day regression action	MONTHLY	TDREGR=
compute summary measure only	MONTHLY	SUMMARY
	QUARTERLY	SUMMARY
modify extreme irregulars prior to trend cycle estimation	MONTHLY	TRENDADJ
	QUARTERLY	TRENDADJ
specify moving average length in trend cycle estimation	MONTHLY	TRENDMA=
	QUARTERLY	TRENDMA=
specify weights for prior trading-day factors	PDWEIGHTS	

PROC X11 Statement

PROC X11 *options;*

The following options can appear in the PROC X11 statement:

DATA= *SAS-data-set*
specifies the input SAS data set used. If it is omitted, the most recently created SAS data set is used.

OUTEXTRAP
adds the extra observations used in ARIMA processing to the output data set.

When ARIMA forecasting/backcasting is requested, extra observations are appended on the ends of the series, and the calculations are carried out on this extended series. The appended observations are not normally written to the OUT= data set. However, if OUTEXTRAP is specified, these extra observations are written to the output data set. If a DATE= variable is specified in the MONTHLY/QUARTERLY statement, the date variable is extrapolated to identify forecasts/backcasts. The OUTEXTRAP option can be abbreviated as OUTEX.

NOPRINT
suppresses any printed output. The NOPRINT option overrides any PRINTOUT=, CHARTS=, or TABLES statement and any output associated with the ARIMA statement.

OUTSPAN= *SAS-data-set*
Specifies the output data set to store the sliding spans analysis results. Tables A1, C18, D10 and D11 for each span are written to this data set. See "OUTSPAN Data Set" later in this chapter for details.

OUTSTB= *SAS-data-set*
Specifies the output data set to store the stable seasonality test results (table D8). All the information in the analysis of variance table associated with the stable seasonality test is contained in the variables written to this data set. See "OUTSTB Data Set" later in this chapter for details.

OUTTDR= *SAS-data-set*

Specifies the output data set to store the trading-day regression results (tables B15 and C15). All the information in the analysis of variance table associated with the trading-day regression is contained in the variables written to this data set. This option is valid only when TDREGR=PRINT, TEST, or ADJUST is specified in the MONTHLY statement. See "OUTTDR Data Set" later in this chapter for details.

YRAHEADOUT

adds one-year-ahead forecast values to the output data set for tables C16, C18 and D10. The original purpose of this option was to avoid recomputation of the seasonal adjustment factors when new data became available. While computing costs were an important factor when the X-11 method was developed, this is no longer the case and this option is obsolete. See "The YRAHEADOUT Option" later in this chapter for details.

ARIMA Statement

ARIMA *options;*

The ARIMA statement applies the X-11-ARIMA method to the series specified in the VAR statement. This method uses an ARIMA model estimated from the original data to extend the series one or more years. The ARIMA statement options control the ARIMA model used and the estimation, forecasting, and printing of this model.

There are two ways of obtaining an ARIMA model to extend the series. A model can be given explicitly with the MODEL= and TRANSFORM= options. Alternatively, the best fitting model from a set of five predefined models is found automatically whenever the MODEL= option is absent. See "Automatic Model Selection" later in this chapter for details.

BACKCAST= *n*

Specifies the number of years to backcast the series. The default is BACKCAST= 0. See "Effect of Backcast and Forecast Length" later in this chapter for details.

CHICR= *value*

specifies the criteria for the significance level for the Box-Ljung chi-square test for lack of fit when testing the five predefined models. The default is CHICR= 0.05. The CHICR= option values must be between 0.01 and 0.90. The hypothesis being tested is that of model adequacy. Nonrejection of the hypothesis is evidence for an adequate model. Making the CHICR= value smaller makes it easier to accept the model. See "Criteria Details" later in this chapter for further details on the CHICR= option.

CONVERGE= *value*

specifies the convergence criterion for the estimation of an ARIMA model. The default value is 0.001. The CONVERGE= value must be positive.

FORECAST= *n*

Specifies the number of years to forecast the series. The default is FORECAST= 1. See "Effect of Backcast and Forecast Length" later in this chapter for details.

MAPECR= *value*

specifies the criteria for the Mean Absolute Percent Error (MAPE) when testing the five predefined models. A small MAPE value is evidence for an adequate model; a large MAPE value results in the model being rejected. The MAPECR= value is the boundary for acceptance/rejection. Thus a larger MAPECR= value would make it easier for a model to pass the criteria. The default is MAPECR= 15. The MAPECR= option values must be between 1 and 100. See "Criteria Details" later in this chapter for further details on the MAPECR= option.

MAXITER= *n*

specifies the maximum number of iterations in the estimation process. MAXITER must be between 1 and 60; the default value is 15.

METHOD= CLS
METHOD= ULS
METHOD= ML

specifies the estimation method. ML requests maximum likelihood, ULS requests unconditional least-squares, and CLS requests conditional least-squares. METHOD=CLS is the default. The maximum likelihood estimates are more expensive to compute than the conditional least-squares estimates. In some cases, however, they may be preferable. For further information on the estimation methods, see "Estimation Details" in Chapter 11, "The ARIMA Procedure."

MODEL= (P=*n1* **Q=***n2* **SP=***n3* **SQ=***n4* **DIF=***n5* **SDIF=***n6* **<NOINT> <CENTER>)**

specifies the ARIMA model. The AR and MA orders are given by P=*n1* and Q=*n2* respectively, while the seasonal AR and MA orders are given by SP=*n3* and SQ=*n4*. The lag corresponding to seasonality is determined by the MONTHLY or QUARTERLY statement. Similarly, differencing and seasonal differencing are given by DIF=*n5* and SDIF=*n6* respectively.

For example

```
   arima model=( p=2 q=1 sp=1 dif=1 sdif=1 );
```

specifies a (2,1,1)(1,1,0)*s* model, where *s*, the seasonality is either 12 (monthly) or 4 (quarterly). More examples of the MODEL= syntax is given in the "Automatic Model Selection" section.

NOINT

suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter μ is omitted.)

CENTER

centers each time series by subtracting its sample mean. The analysis is done on the centered data. Later, when forecasts are generated, the mean is added back. Note that centering is done after differencing. The CENTER option is normally used in conjunction with the NOCONSTANT option of the ESTIMATE statement.

For example, to fit an AR(1) model on the centered data without an intercept, use the following ARIMA statement.

```
   arima model=( p=1 center noint );
```

NOPRINT

suppresses the normal printout generated by the ARIMA statement. Note that the effect of NOPRINT on the ARIMA statement is different from NOPRINT on the PROC statement, since the former only affects ARIMA output.

OVDIFCR= *value*

specifies the criteria for the over-differencing test when testing the five predefined models. When the MA parameters in one of these models sum to a number close to 1.0, this is an indication of over-parameterization and the model is rejected. The OVDIFCR= value is the boundary for this rejection; values greater than this value fail the over-differencing test. A larger OVDIFCR= value would make it easier for a model to pass the criteria. The default is OVDIFCR= 0.90. The OVDIFCR= option values must be between 0.80 and 0.99. See "Criteria Details" later in this chapter for further details on the OVDIFCR= option.

PRINTALL

provides the same output as the default printing for all models fit and, in addition, prints an estimation summary and chi-square statistics for each model fit. See "Printed Output" later in this chapter for details.

PRINTFP

prints the results for the initial pass of X11 made to exclude trading-day effects. This option has an effect only when the TDREGR= option specifies ADJUST, TEST, or PRINT. In these cases, an initial pass of the standard X11 method is required to get rid of calendar effects before doing any ARIMA estimation. Usually this first pass is not of interest, and by default no tables are printed. However, specifying PRINTFP on the ARIMA statement causes any tables printed in the final pass to also be printed for this initial pass.

TRANSFORM= (LOG) | LOG
TRANSFORM= (*constant ** power* **)**

The ARIMA statement in PROC X11 allows certain transformations on the series before estimation. The specified transformation is applied only to a user-specified model. If TRANSFORM= is specified without a MODEL=, the transformation request is ignored and a warning is printed.

The LOG transformation requests that the natural log of the series be used for estimation. The resulting forecasted values are transformed back to the original scale.

A general power transformation of the form $X_t \rightarrow (X_t + a)^b$ is obtained by specifying

```
   transform= ( a ** b )
```

If the constant *a* is not specified, it is assumed to be zero. The specified ARIMA model is then estimated using the transformed series. The resulting forecasted values are transformed back to the original scale.

BY Statement

BY *variables;*

A BY statement can be used with PROC X11 to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input DATA= data set to be sorted in order of the BY variables.

ID Statement

ID *variables;*

If you are creating an output data set, use the ID statement to put values of the ID variables, in addition to the table values, into the output data set. The ID statement has no effect when an output data set is not created. If the DATE= variable is specified in the MONTHLY or QUARTERLY statement, this variable is included automatically in the OUTPUT data set. If no DATE= variable is specified, the variable _DATE_ is added.

The date variable (or _DATE_) values outside the range of the actual data (from ARIMA forecasting, backcasting, or from YRAHEADOUT) are extrapolated, while all other ID variables are missing.

MACURVES Statement

MACURVES *month=option ...;*

The MACURVES statement specifies the length of the moving average curves for estimating the seasonal factors for any month. This statement can be used only with monthly time series data.

The *month=option* specifications consist of the month name (or the first three letters of the month name), an equal sign, and one of the following option values.

’3’	specifies a three-term moving average for the month
’3X3’	specifies a three-by-three moving average
’3X5’	specifies a three-by-five moving average
’3X9’	specifies a three-by-nine moving average
STABLE	specifies a stable seasonal factor (average of all values for the month)

For example, the statement

```
   macurves jan='3' feb='3x3' march='3x5' april='3x9';
```

uses a three-term moving average to estimate seasonal factors for January, a 3x3 (a three-term moving average of a three term moving average) for February, a 3x5 (a three-term moving average of a five-term moving average) for March, and a 3x9 (a three-term moving average of a nine-term moving average) for April.

The numeric values used for the weights of the various moving averages and a discussion of the derivation of these weights are given in U.S. Bureau of Census, 1967. A general discussion of moving average weights is given in Dagum, 1985.

If the specification for a month is omitted, the X11 procedure uses a three-by-three moving average for the first estimate of each iteration and a three-by-five average for the second estimate.

MONTHLY Statement

MONTHLY *options;*

The MONTHLY statement must be used when the input data to PROC X11 is a monthly time series. The MONTHLY statement specifies options that determine the computations performed by PROC X11 and what is included in its output. Either the DATE= or START= option must be used.

The following options can appear in the MONTHLY statement:

ADDITIVE

performs additive adjustments. If the ADDITIVE option is omitted, PROC X11 performs multiplicative adjustments.

CHARTS= STANDARD
CHARTS= FULL
CHARTS= NONE

specifies the charts produced by the procedure. The default is CHARTS=STANDARD, which specifies 12 monthly seasonal charts and a trend cycle chart. If you specify CHARTS=FULL (or CHARTS=ALL), the procedure prints additional charts of irregular and seasonal factors. To print no charts, specify CHARTS=NONE.

The TABLES statement can also be used to specify particular monthly charts to be printed. If no CHARTS= is given, and a TABLES statement is given, the TABLES statement overrides the default value of CHARTS=STANDARD; that is, no charts (or tables) are printed except those specified in the TABLES statement. However, if both the CHARTS= option and a TABLES statement are given, the charts corresponding to the CHARTS= option and those requested by the TABLES statement are printed.

For example, suppose you wanted only charts G1, the final seasonally adjusted series and trend cycle, and G4, the final irregular and final modified irregular series. You would specify the following statements.

```
monthly date=date;
tables g1 g4;
```

DATE= *variable*

specifies a variable that gives the date for each observation. The starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS date values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly. This variable is automatically added to the OUTPUT= data set if one is requested and extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

The DATE= option and the START= and END= options can be used in combination to subset a series for processing. For example, suppose you have 12 years of monthly data (144 observations, no missing values) beginning in January, 1970 and ending in December, 1981, and you only wanted to seasonally adjust six years beginning in January of 1974. Specifying

```
monthly date=date start=jan1974 end=dec1979;
```

would seasonally adjust only this subset of the data. If, instead, you wanted to adjust the last eight years of data, only the START= is needed:

```
monthly date=date start=jan1974;
```

END= *mmmyyyy*

specifies that only the part of the input series ending with the month and year given be adjusted (for example, END=DEC1970). See the DATE=*variable* option for using the START= and END= options to subset a series for processing.

EXCLUDE= *value*

excludes from the trading-day regression any irregular values that are more than *value* standard deviations from the mean. The EXCLUDE=value must be between .1 and 9.9, with the default value being 2.5.

FULLWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values less than the FULLWEIGHT= *value* (in standard deviation units) are assigned full weights of 1, values that fall between the ZEROWEIGHT= and FULLWEIGHT= limits are assigned weights linearly graduated between 0 and 1, and values greater than the ZEROWEIGHT= limit are assigned a weight of 0.

For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The FULLWEIGHT=value must be between .1 and 9.9 but must be less than the ZEROWEIGHT=value. The default is FULLWEIGHT=1.5.

LENGTH

includes length-of-month allowance in computing trading-day factors. If this option is omitted, length-of-month allowances are included with the seasonal factors.

NDEC= *n*

specifies the number of decimal places shown on the printed tables on the listing. This option has no effect on the precision of the variables values in the output data set.

PMFACTOR= *variable*

specifies a variable containing the prior monthly factors. Use this option if you have previous knowledge of monthly adjustment factors. The PMFACTOR= option can be used to:

- adjust the level of all or part of a series with discontinuities
- adjust for the influence of holidays that fall on different dates from year to year, such as the effect of Easter on certain retail sales
- adjust for unreasonable weather influence on series, such as housing starts
- adjust for changing starting dates of fiscal years (for budget series) or model years (for automobiles)
- adjust for temporary dislocating events, such as strikes

See "Prior Daily Weights and Trading-Day Regression" in the "Details" section later in this chapter for details and examples using the PMFACTOR= option.

PRINTOUT= STANDARD | LONG | FULL | NONE

specifies the tables to be printed by the procedure. If the PRINTOUT=STANDARD option is specified, between 17 and 27 tables are printed, depending on the other options that are specified. PRINTOUT=LONG prints between 27 and 39 tables, and PRINTOUT=FULL prints between 44 and 59 tables. Specifying PRINTOUT=NONE results in no tables being printed; however, charts are still printed. The default is PRINTOUT=STANDARD.

The TABLES statement can also be used to specify particular monthly tables to be printed. If no PRINTOUT= is given, and a TABLES statement is given, the TABLES statement overrides the default value of PRINTOUT=STANDARD; that is, no tables (or charts) are printed except those given in the TABLES statement. However, if both the PRINTOUT= option and a TABLES statement are given, the tables corresponding to the PRINTOUT= option and those requested by the TABLES statement are printed.

START= *mmmyyyy*

adjusts only the part of the input series starting with the specified month and year. When the DATE= option is not used, the START= option gives the year and month of the first input observation. For example, START=JAN1966. START= must be specified if DATE= is not given. If START= is specified (and no DATE= option is given), and an OUT= data set is requested, a variable named _DATE_ is added to the data set, giving the date value for each observation. See the DATE= *variable* option for using the START= and END= options to subset a series.

SUMMARY

specifies that the data are already seasonally adjusted and the procedure is to produce summary measures. If the SUMMARY option is omitted, the X11 procedure performs seasonal adjustment of the input data before calculating summary measures.

TDCOMPUTE= *year*

uses the part of the input series beginning with January of the specified year to derive trading-day weights. If this option is omitted, the entire series is used.

TDREGR= NONE | PRINT | ADJUST | TEST

specifies the treatment of trading-day regression. The value NONE omits the computation of the trading-day regression. The value PRINT computes and prints the trading-day regressions but does not adjust the series. The value ADJUST computes and prints the trading-day regression and adjusts the irregular components to obtain preliminary weights. The value TEST adjusts the final series if the trading-day regression estimates explain significant variation on the basis of an *F* test (or residual trading-day variation if prior weights are used). The default is TDREGR=NONE.

See "Prior Daily Weights and Trading-Day Regression" in the "Details" section later in this chapter for details and examples using the TDREGR= option.

If ARIMA processing is requested, any value of TDREGR other than the default TDREGR=NONE will cause PROC X11 to perform an initial pass (see the "Details" section and the PRINTFP option).

The signifigance level reported Table C15 should be viewed with caution. The dependent variable in the trading day regression is the irregular component formed by an averaging operation. This induces a correlation in the dependent variable and hence in the residuals from which which the F-test is computed. Hence the distribution of the trading day regression F-statistics differs from an exact F; see Cleveland and Devlin, 1980 for details.

TRENDADJ

modifies extreme irregular values prior to computing the trend cycle estimates in the first iteration. If the TRENDADJ option is omitted, the trend cycle is computed without modifications for extremes.

TRENDMA= 9 | 13 | 23.

specifies the number of terms in the moving average to be used by the procedure in estimating the variable trend cycle component. The value of the TRENDMA= option must be 9, 13, or 23. If the TRENDMA= option is omitted, the procedure selects an appropriate moving average. For information concerning the number of terms in the moving average, see U.S. Bureau of the Census (1967).

ZEROWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values beyond the standard deviation limit specified in the ZEROWEIGHT= option are assigned zero weights. Values that fall between the two limits (ZEROWEIGHT= and FULLWEIGHT=) are assigned weights linearly graduated between 0 and 1. For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The ZEROWEIGHT=value must be between .1 and 9.9 but must be greater than the FULLWEIGHT=value. The default is ZEROWEIGHT=2.5.

The ZEROWEIGHT option can be used in conjunction with the FULLWEIGHT= option to adjust outliers from a monthly or quarterly series. See Example 31.3 later in this chapter for an illustration of this use.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set tablename=var1 var2 ... ;*

The OUTPUT statement creates an output data set containing specified tables. The data set is named by the OUT= option.

OUT= *SAS-data-set*

If OUT= is omitted, the SAS System names the new data set using the DATA*n* convention.

For each table to be included in the output data set, write the X11 table identification keyword, an equal sign, and a list of new variable names.

tablename = var1 var2 ...

The *tablename* keywords that can be used in the OUTPUT statement are listed in the "Printed Output" section on page 1899. The following is an example of a VAR and OUTPUT statement.

```
var z1 z2 z3;
output out=out_x11  b1=s  d11=w x y;
```

The variable s contains the table B1 values for the variable z1, while the table D11 values for variables z1, z2, and z3 are contained in variables w, x, and y respectively. As this example shows, the list of variables following a *tablename*= keyword can be shorter than the VAR variable list.

In addition to the variables named by *tablename*=*var1 var2* ... , the ID variables, and BY variables, the output data set contains a date identifier variable. If the DATE= option is given in the MONTHLY or QUARTERLY statement, the DATE= variable is the date identifier. If no DATE= is given, a variable named _DATE_ is the date identifier.

PDWEIGHTS Statement

PDWEIGHTS *day=w ... ;*

The PDWEIGHTS statement can be used to specify one to seven daily weights. The statement can only be used with monthly series using the multiplicative model. These weights are used to compute prior trading-day factors, which are then used to adjust the original series prior to the seasonal adjustment process. Only relative weights are needed; the X11 procedure adjusts the weights so that they sum to 7.0. The weights can also be corrected by the procedure on the basis of estimates of trading-day variation from the input data.

See "Prior Daily Weights and Trading-Day Regression" in the "Details" section later in this chapter for details and examples using the PDWEIGHTS statement.

Each *day*=*w* option specifies a weight (*w*) for the named day. The *day* can be any day, Sunday through Saturday. The *day* keyword can be the full spelling of the day, or the three letter abbreviation. For example, SATURDAY=1.0 and SAT=1.0 are both valid. The weights *w* must be a numeric value between 0.0 and 10.0.

The following is an example of a PDWEIGHTS statement.

```
   pdweights sun=.2 mon=.9 tue=1 wed=1 thu=1 fri=.8 sat=.3;
```

Any number of days can be specified with one PDWEIGHTS statement. The default weight value for any day that is not specified is 0. If you do not use a PDWEIGHTS statement, the program computes daily weights if TDREGR=ADJUST is specified. Refer to U.S. Bureau of the Census (1967) for details.

QUARTERLY Statement

QUARTERLY *options;*

The QUARTERLY statement must be used when the input data are quarterly time series. This statement includes options that determine the computations performed by the procedure and what is in the printed output. The DATE= option or the START= option must be used.

The following options can appear in the QUARTERLY statement.

ADDITIVE

performs additive adjustments. If this option is omitted, the procedure performs multiplicative adjustments.

CHARTS= STANDARD
CHARTS= FULL
CHARTS= NONE

specifies the charts to be produced by the procedure. The default value is CHARTS=STANDARD, which specifies four quarterly seasonal charts and a trend cycle chart. If you specify CHARTS=FULL (or CHARTS=ALL), the procedure prints additional charts of irregular and seasonal factors. To print no charts, specify CHARTS=NONE. The TABLES statement can also be used to specify particular charts to be printed. The presence of a TABLES statement overrides the default value of CHARTS=STANDARD, that is, if a TABLES statement is specified, and no CHARTS=option is specified, no charts (nor tables) are printed except those given in the TABLES statement. However, if both the CHARTS= option and a TABLES statement are given, the charts corresponding to the CHARTS= option and those requested by the TABLES statement are printed.

For example, suppose you only wanted charts G1, the final seasonally adjusted series and trend cycle, and G4, the final irregular and final modified irregular series. This is accomplished by specifying the following statements.

```
quarterly date=date;
tables g1 g4;
```

DATE= *variable*

specifies a variable that gives the date for each observation. The starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS date values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly. This variable is automatically added to the OUTPUT= data set if one is requested, and extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

The DATE= option and the START= and END= options can be used in combination to subset a series for processing. For example, suppose you have a series with 10 years of quarterly data (40 observations, no missing values) beginning in ’1970Q1’ and ending in ’1979Q4’, and you only want to seasonally adjust four years beginning in 1974Q1 and ending in 1977Q4. Specifying

```
quarterly date=variable start='1974q1' end='1977q4';
```

seasonally adjusts only this subset of the data. If, instead, you wanted to adjust the last six years of data, only the START= is needed:

```
quarterly date=variable start='1974q1';
```

END= *‘yyyyQq’*

specifies that only the part of the input series ending with the quarter and year given be adjusted (for example, END=’1973Q4’). The specification must be enclosed in quotes and *q* must be 1, 2, 3, or 4. See the DATE= *variable* option for using the START= and END= options to subset a series.

FULLWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values less than the FULLWEIGHT= value (in standard deviation units) are assigned full weights of 1, values that fall between the ZEROWEIGHT= and FULLWEIGHT= limits are assigned weights linearly graduated between 0 and 1, and values greater than the ZEROWEIGHT= limit are assigned a weight of 0.

For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The default is FULLWEIGHT=1.5.

NDEC= *n*

specifies the number of decimal places shown on the output tables. This option has no effect on the precision of the variables in the output data set.

PRINTOUT= STANDARD
PRINTOUT= LONG
PRINTOUT= FULL
PRINTOUT= NONE

specifies the tables to print. If PRINTOUT=STANDARD is specified, between 17 and 27 tables are printed, depending on the other options that are specified. PRINTOUT=LONG prints between 27 and 39 tables, and PRINTOUT=FULL prints between 44 and 59 tables. Specifying PRINTOUT=NONE results in no tables being printed. The default is PRINTOUT=STANDARD.

The TABLES statement can also specify particular quarterly tables to be printed. If no PRINTOUT= is given, and a TABLES statement is given, the TABLES statement overrides the default value of PRINTOUT=STANDARD; that is, no tables (or charts) are printed except those given in the TABLES statement. However, if both the PRINTOUT= option and a TABLES statement are given, the tables corresponding to the PRINTOUT= option and those requested by the TABLES statement are printed.

START= *'yyyyQq'*

adjusts only the part of the input series starting with the quarter and year given. When the DATE= option is not used, the START= option gives the year and quarter of the first input observation (for example, START='1967Q1'). The specification must be enclosed in quotes, and *q* must be 1, 2, 3, or 4. START= must be specified if the DATE= option is not given. If START= is specified (and no DATE= is given), and an OUTPUT= data set is requested, a variable named _DATE_ is added to the data set, giving the date value for a given observation. See the DATE= option for using the START= and END= options to subset a series.

SUMMARY

specifies that the input is already seasonally adjusted and that the procedure is to produce summary measures. If this option is omitted, the procedure performs seasonal adjustment of the input data before calculating summary measures.

TRENDADJ

modifies extreme irregular values prior to computing the trend cycle estimates. If this option is omitted, the trend cycle is computed without modification for extremes.

ZEROWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values beyond the standard deviation limit specified in the ZEROWEIGHT= option are assigned zero weights. Values that fall between the two limits (ZEROWEIGHT= and FULLWEIGHT=) are assigned weights linearly graduated between 0 and 1. For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The default is ZEROWEIGHT=2.5.

The ZEROWEIGHT option can be used in conjunction with the FULLWEIGHT= option to adjust outliers from a monthly or quarterly series. See Example 31.3 later in this chapter for an illustration of this use.

SSPAN Statement

SSPAN *options ;*

The SSPAN statement applies sliding spans analysis to determine the suitability of seasonal adjustment for an economic series.

The following options can appear in the SSPAN Statement:

NDEC= *n*
specifies the number of decimal places shown on selected Sliding Span reports. This option has no effect on the precision of the variables values in the OUTSPAN output data set.

CUTOFF= *value*
gives the percentage value for determining an excessive difference within a span for the seasonal factors, the seasonally adjusted series, and month-to-month and year-to-years differences in the seasonally adjusted series. The default value is 3.0. The use of the CUTOFF=value in determining the maximum percent difference (MPD) is described in "Computational Details" later in this section. Caution should be used in changing the default CUTOFF=value. The empirical threshold ranges found by the Census Bureau no longer apply when value is changed.

TDCUTOFF= *value*
gives the percentage value for determining an excessive difference within a span for the trading day factors. The default value is 2.0. The use of the TDCUTOFF=value in determining the maximum percent difference (MPD) is described in "Computational Details" later in this section. Caution should be used in changing the default TDCUTOFF=value. The empirical threshold ranges found by the Census Bureau no longer apply when value is changed.

NOPRINT
suppresses all sliding spans reports.

PRINT
prints the summary sliding spans reports S 0 - S 6.E.

PRINTALL
prints the summary sliding spans reports S 0 - S 6.E, along with detail reports S 7.A - S 7.E.

TABLES Statement

TABLES *tablenames;*

The TABLES statement prints the tables specified in addition to the tables that are printed as a result of the PRINTOUT= option in the MONTHLY or QUARTERLY statement. Table names are listed in Table 31.3 later in this chapter.

To print only selected tables, omit the PRINTOUT= option in the MONTHLY or QUARTERLY statement and list the tables to be printed on the TABLES statement. For example, to print only the final seasonal factors and final seasonally adjusted series, use the statement

```
tables d10 d11;
```

VAR Statement

VAR *variables;*

The VAR statement is used to specify the variables in the input data set that are to be analyzed by the procedure. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are analyzed except those appearing in a BY or ID statement or the variable named in the DATE= option in the MONTHLY or QUARTERLY statement.

Details

Historical Development of X-11

This section briefly describes the historical development of the standard X-11 seasonal adjustment method and the later development of the X-11-ARIMA method. Most of the following discussion is based on a comprehensive article by Bell and Hillmer (1984), which describes the history of X-11 and the justification of using seasonal adjustment methods, such as X-11, given the current availability of time series software. For further discussions on statistical problems associated with the X-11 method, refer to Ghysels (1990).

Seasonal adjustment methods began development in the 1920s and 1930s before there were suitable analytic models available and before electronic computing devices were developed. The lack of any suitable model led to methods that worked the same for any series, that is, methods that were not model-based and that could be applied to any series. Experience with economic series had shown that a given mathematical form could adequately represent a time series only for a fixed length; as more data was added, the model became inadequate. This suggested an approach using moving averages.

The basic method was to break up an economic time series into long-term trend, long-term cyclical movements, seasonal movements, and irregular fluctuations.

Early investigators found that it was not possible to uniquely decompose the trend and cycle components. Thus, these two were grouped together; the resulting component is usually referred to as the "trend cycle component."

It was also found that estimating seasonal components in the presence of trend produced biased estimates of the seasonal components, but, at the same time, estimating trend in the presence of seasonality was difficult. This eventually lead to the iterative approach used in the X-11 method.

Two other problems were encountered by early investigators. First, some economic series appears to have changing or evolving seasonality. Secondly, moving averages were very sensitive to extreme values. The estimation method used in the X-11 method allows for evolving seasonal components. For the second problem, the X-11 method uses repeated adjustment of extreme values.

All of these problems encountered in the early investigation of seasonal adjustment methods suggested the use of moving averages in estimating components. Even with the use of moving averages instead of a model-based method, massive amounts of hand calculations were required. Only a small number of series could be adjusted, and little experimentation could be done to evaluate variations on the method.

With the advent of electronic computing in the 1950s, work on seasonal adjustment methods proceeded rapidly. These methods still used the framework previously described; variants of these basic methods could now be easily tested against a large number of series.

Much of the work was done by Julian Shiskin and others at the U.S. Bureau of the Census beginning in 1954 and culminated after a number of variants into the *X-11 Variant of the Census Method II Seasonal Adjustment Program*, which PROC X11 implements.

References for this work during this period include Shiskin and Eisenpress (1957), Shiskin (1958), and Marris (1960). The authoritative documentation for the X-11 Variant is in U.S. Bureau of the Census (1967). This document is not equivalent to a program specification; however the FORTRAN code implementing the X-11 Variant is in the public domain. A less detailed description of the X-11 Variant is given in U.S. Bureau of the Census (1969).

Development of the X-11-ARIMA Method

The X-11 method uses symmetric moving averages in estimating the various components. At the end of the series, however, these symmetric weights cannot be applied. Either asymmetric weights have to be used, or some method of extending the series must be found.

While various methods of extending a series have been proposed, the most important method to date has been the X-11-ARIMA method developed at Statistics Canada. This method uses Box-Jenkins ARIMA models to extend the series.

The Time Series Research and Analysis Division of Statistic Canada investigated 174 Canadian economic series and found five ARIMA models out of twelve that fit the majority of series well and reduced revisions for the most recent months. References giving details of various aspects of the X-11-ARIMA methodology include Dagum (1980, 1982a, 1982b, 1983, 1988), Laniel (1985), Lothian and Morry (1978a), and Huot,Chui, Higginson, and Gait (1986).

Differences Between X11ARIMA/88 and PROC X11

The original implementation of the X-11-ARIMA method was by Statistics Canada in 1980 (Dagum, 1980; X11ARIMA/80), with later changes and enhancements made in 1988 (Dagum, 1988; X11ARIMA/88). The calculations performed by PROC X11 differ from those in X11ARIMA/88, which will result in differences in the final component estimates provided by these implementations.

There are three areas where Statistic Canada made changes to the original X-11 seasonal adjustment method in developing X11ARIMA/80 (refer to Monsell, 1984). These are (a) selection of extreme values, (b) replacement of extreme values, and (c) generation of seasonal and trend cycle weights.

These changes have not been implemented in the current version of PROC X11. Thus the procedure produces identical results with previous versions of PROC X11 in the absence of an ARIMA statement.

Additional differences can result from the ARIMA estimation. X11ARIMA/88 uses Conditional Least Squares (CLS), while CLS, Unconditional Least Squares (ULS) and Maximum Likelihood (ML) are all available in PROC X11 by using the METHOD= option on the ARIMA statement. Generally, parameters estimates will differ for the different methods.

Implementation of the X-11 Seasonal Adjustment Method

The following steps describe the analysis of a monthly time series using multiplicative adjustments. Additional steps used by the X-11-ARIMA method are also indicated. Equivalent descriptions apply for an additive model by replacing *divide* by *subtract* where applicable.

In the multiplicative adjustment, the original series O_t is assumed to be of the form

$$O_t = C_t S_t I_t P_t D_t,$$

where C_t is the trend cycle component, S_t is the seasonal component, I_t is the irregular component, P_t is the prior monthly factors component and D_t is the trading-day component.

The trading-day component can be further factored as

$$D_t = D_{r,t} D_{tr,t},$$

where $D_{tr,t}$ are the trading-day factors derived from the prior daily weights, and $D_{r,t}$ are the residual trading-day factors estimated from the trading-day regression.

Additional Steps When Using the X-11-ARIMA Method

The X-11-ARIMA method consists of extending a given series by an ARIMA model and applying the usual X-11 seasonal adjustment method to this extended series. Thus in the simplest case in which there are no prior factors or calendar effects in the series, the ARIMA model selection, estimation and forecasting is first performed, and the resulting extended series goes through the standard X-11 steps described below.

If prior factor or calendar effects are present, they must be eliminated from the series before the ARIMA estimation is done because these effects are not stochastic.

Prior factors, if present, are removed first. Calendar effects represented by prior daily weights are then removed. If there are no further calendar effects, the adjusted series is extended by the ARIMA model, and this extended series goes through the standard X-11 steps without repeating the removal of prior factors and calendar effects from prior daily weights.

If further calendar effects are present, a trading-day regression must be performed. In this case it is necessary to go through an initial pass of the X-11 steps to obtain a final trading-day adjustment. In this initial pass, the series, adjusted for prior factors and prior daily weights, goes through the standard X-11 steps. At the conclusion of these steps, a final series adjusted for prior factors and all calendar effects is available. This adjusted series is then extended by the ARIMA model, and this extended series goes through the standard X-11 steps again, without repeating the removal of prior factors and calendar effects from prior daily weights and trading day regression.

The Standard X-11 Seasonal Adjustment Method

The following steps comprise the standard X-11 seasonal adjustment method. These steps are applied to the original data or the original data extended by an ARIMA model.

1. In step 1, the data are read, ignoring missing values until the first nonmissing value is found. If prior monthly factors are present, the procedure reads prior monthly P_t factors and divides them into the original series to obtain $O_t/P_t = C_t S_t I_t D_{tr,t} D_{r,t}$.

   Seven daily weights can be specified to develop monthly factors to adjust the series for trading-day variation, $D_{tr,t}$; these factors are then divided into the original or prior adjusted series to obtain $C_t S_t I_t D_{r,t}$.

2. In steps 2, 3, and 4, three iterations are performed, each of which provides estimates of the seasonal S_t, trading-day $D_{r,t}$, trend cycle C_t, and irregular components I_t. Each iteration refines estimates of the extreme values in the irregular components. After extreme values are identified and modified, final estimates of the seasonal component, seasonally adjusted series, trend cycle, and irregular components are produced. Step 2 consists of three substeps:

   (a) During the first iteration, a centered, 12-term moving average is applied to the original series O_t to provide a preliminary estimate $\hat{C}_t$ of the trend cycle curve C_t. This moving average combines 13 (a 2 term moving average of a 12-term moving average) consecutive monthly values, removing the S_t and I_t. Next, it obtains a preliminary estimate $\widehat{S_t I_t}$ by

   $$\widehat{S_t I_t} = \frac{O_t}{\hat{C}_t}$$

   (b) A moving average is then applied to the $\widehat{S_t I_t}$ to obtain an estimate $\hat{S}_t$ of the seasonal factors. $\widehat{S_t I_t}$ is then divided by this estimate to obtain an estimate $\hat{I}_t$ of the irregular component. Next, a moving standard deviation is calculated from the irregular component and is used in assigning a weight to each monthly value for measuring its degree of extremeness.

These weights are used to modify extreme values in $\widehat{S_t I_t}$. New seasonal factors are estimated by applying a moving average to the modified value of $\widehat{S_t I_t}$. A preliminary seasonally adjusted series is obtained by dividing the original series by these new seasonal factors. A second estimate of the trend cycle is obtained by applying a weighted moving average to this seasonally adjusted series.

(c) The same process is used to obtain second estimates of the seasonally adjusted series and improved estimates of the irregular component. This irregular component is again modified for extreme values and then used to provide estimates of trading-day factors and refined weights for the identification of extreme values.

3. Using the same computations, a second iteration is performed on the original series that has been adjusted by the trading-day factors and irregular weights developed in the first iteration. The second iteration produces final estimates of the trading-day factors and irregular weights.
4. A third and final iteration is performed using the original series that has been adjusted for trading-day factors and irregular weights computed during the second iteration. During the third iteration, PROC X11 develops final estimates of seasonal factors, the seasonally adjusted series, the trend cycle, and the irregular components. The procedure computes summary measures of variation and produces a moving average of the final adjusted series.

Sliding Spans Analysis

The motivation for sliding spans analysis is to answer the question "When is a economic series unsuitable for seasonal adjustment ?". There are a number of past attempts to answer this question: stable seasonality F-test; moving seasonality F-test, Q-statistics and others.

Sliding spans analysis attempts to quantify the stability of the seasonal adjustment process, and hence quantify the suitability of seasonal adjustment for a given series.

It is based on a very simple idea: for a stable series, deleting a small number of observations should not result in greatly different component estimates compared with the original, full series. Conversely, if deleting a small number of observations results in drastically different estimates, the series is unstable. For example, a drastic difference in the seasonal factors (Table D10) might result from a dominating irregular component, or sudden changes in the seasonally component. When the seasonal component estimates of a series is unstable in this manner, they have little meaning and the series is likely to be unsuitable for seasonal adjustment.

Sliding spans analysis, developed at the Statistical Research Division of the U.S. Census Bureau (see Findley, et al., 1990, and Findley and Monsell, 1986), performs a repeated seasonal adjustment on subsets or spans of the full series. In particular, an initial span of the data, typically eight years in length, is seasonally adjusted, and the tables C18, the trading day factors (if trading day regression performed), D10, the seasonal factors, and D11, the seasonally adjusted series are retained for further processing. Next, one year of data is deleted from the beginning of the initial span and one year of data is added. This new span is seasonally adjusted as before, with the same tables retained. This process continues until the end of the data is reached. The beginning and ending dates of the spans are such that the last observation in the original data is also the last observation in the last span. This is discussed in more detail below.

The following notation for the components or differences computed in the sliding spans analysis follows Findley et al., 1990. The meaning for the symbol X_t(k) is component X in month (or quarter) t, computed from data in the k-th span. These components are now defined.

Seasonal Factors (Table D10): S_t(k)

Trading Day Factor (Table C18): TD_t(k)

Seasonally Adjust Data (Table D11): SA_t(k)

Month-to-month changes in the Seasonally Adjust Data: MM_t(k)

Year-to-Year changes in the Seasonally Adjust Data: YY_t(k)

The key measure is the maximum percent difference across spans. For example, consider a series beginning in JAN72, ending in DEC84, and having four spans, each of length 8 years (see Figure 1. in Findley et al., 1990, page 346). Consider S_t(k) the seasonal factor (table D10) for month t for span k, and let N_t denote the number of spans containing month t, i.e.,

$$N_t = \{k : \ span\ k\ contains\ month\ t\}$$

In the middle years of the series there is overlap of all four spans and N_t will be 4. The last year of the series will have but one span, while the beginning can have 1 or 0 spans depending on the original length.

Since we are interested in how much the seasonal factors vary for a given month across the spans, a natural quantity to consider is

$$max_{k \epsilon N_t} S_t(k) - min_{k \epsilon N_t} S_t(k)$$

In the case of the multiplicative model, it is useful to compute a percent difference; define the maximum percent difference (MPD) at time t as

$$MPD_t = \frac{max_{k \epsilon N_t} S_t(k) - min_{k \epsilon N_t} S_t(k)}{min_{k \epsilon N_t} S_t(k)}$$

The seasonal factor for month t is then unreliable if MPD_t is large. While no exact significance level can be computed for this statistic, empirical levels have been established by considering over 500 economic series (see Findley, et al. 1990 and Findley and Monsell, 1986). For these series it was found that for four spans, stable series typically had less than 15% of the MPD values exceeding 3.0%, while in marginally stable series, between 15% and 25% of the MPD values exceeded 3.0%. A series in which 25% or more of the MPD values exceeded 3.0% is almost always unstable.

While these empirical values cannot be considered an exact significance level, they provide a useful empirical basis for deciding if a series is suitable for seasonal adjustment. These percentage values are shifted down when less than four spans are used.

Computation Details for Sliding Spans Analysis

Length and Number of Spans

The algorithm for determining the length and number of spans for a given series was developed at the U.S. Bureau of the Census, Statistical Research Division. A summary of this algorithm is as follows.

First, an initial length based on MACURVE specification is determined, then the maximum number of spans possible using this length is determined. If this maximum number exceed four, set the number of spans to four. If this maximum number is one or zero, there is not enough observations to perform the sliding spans analysis. In this case a note is written to the log and the sliding spans analysis is skipped for this variable.

If the maximum number of spans is two or three, the actual number of spans used is set equal to this maximum. Finally, the length is adjusted so that the spans begin in January (or the first quarter) of the beginning year of the span.

The remaining part of this section gives the computation formulas for the maximum percent difference (MPD) calculations along with the threshold regions.

Seasonal Factors (Table D10): $S_t(k)$

For the additive model, the MPD is defined as

$$max_{k \epsilon N_t} S_t(k) - min_{k \epsilon N_t} S_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k \epsilon N_t} S_t(k) - min_{k \epsilon N_t} S_t(k)}{min_{k \epsilon N_t} S_t(k)}$$

A series for which less than 15% of the MPD values of D10 exceed 3.0% is stable; between 15% and 25% is marginally stable; and greater than 25% unstable. Span reports S 2.A - S 2.C give the various breakdowns for the number of times the MPD exceeded these levels.

Trading Day Factor (Table C18): TD$_t$(k)

For the additive model, the MPD is defined as

$$max_{k\epsilon N_t} TD_t(k) - min_{k\epsilon N_t} TD_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k\epsilon N_t} TD_t(k) - min_{k\epsilon N_t} TD_t(k)}{min_{k\epsilon N_t} TD_t(k)}$$

The Census Bureau currently gives no recommendation concerning MPD thresholds for the Trading Day factors. Span reports S 3.A - S 3.C give the various breakdowns for MPD thresholds. When TDREGR=NONE is specified, no trading day computations are done, hence this table is skipped.

Seasonally Adjust Data (Table D11): SA$_t$(k)

For the additive model, the MPD is defined as

$$max_{k\epsilon N_t} SA_t(k) - min_{k\epsilon N_t} SA_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k\epsilon N_t} SA_t(k) - min_{k\epsilon N_t} SA_t(k)}{min_{k\epsilon N_t} SA_t(k)}$$

A series for which less than 15% of the MPD values of D11 exceed 3.0% is stable; between 15% and 25% is marginally stable; and greater than 25% unstable. Span reports S 4.A - S 4.C give the various breakdowns for the number of times the MPD exceeded these levels.

Month-to-Month Changes in the Seasonally Adjust Data: MM$_t$(k)

Some additional notation is needed for the month-to-month and year-to-year differences. Define N1$_t$ by

$$N1_t = \{k : \; span \; k \; contains \; month \; t \; and \; t-1\}$$

For the additive model the month-to-month change for span k is defined by

$$MM_t(k) = SA_t - SA_{t-1}$$

while for the multiplicative model

$$MM_t(k) = \frac{SA_t - SA_{t-1}}{SA_{t-1}},$$

Since this quantity is already in percentage form, the MPD for both the additive and multiplicative model is defined by

$$MPD_t = max_{k \epsilon N1_t} MM_t(k) - min_{k \epsilon N1_t} MM_t(k)$$

The current recommendation of the Census Bureau is that if 35% or more of the MPD values of the month-to-month differences of D11 exceed 3.0% then the series is usually not stable. 40% exceeding this level clearly marks an unstable series. Span reports S 5.A.1 - S 5.C give the various breakdowns for number of times the MPD exceeds these levels.

Year-to-Year Changes in the Seasonally Adjust Data: YY$_t$(k)

First define $N12_t$ by

$$N12_t = \{k : \; span\ k\ contains\ month\ t\ and\ t-12\}$$

(appropriate changes in notation for a quarterly series are obvious.)

For the additive model the month-to-month change for span k is defined by

$$YY_t(k) = SA_t - SA_{t-12}$$

while for the multiplicative model

$$YY_t(k) = \frac{SA_t - SA_{t-12}}{SA_{t-12}},$$

Since this quantity is already in percentage form, the MPD for both the additive and multiplicative model is defined by

$$MPD_t = max_{k \epsilon N1_t} YY_t(k) - min_{k \epsilon N1_t} YY_t(k)$$

The current recommendation of the Census Bureau is that if 10% or more of the MPD values of the month-to-month differences of D11 exceed 3.0% then the series is usually not stable. Span reports S 6.A - S 6.C give the various breakdowns for the number of times the MPD exceeds these levels.

Data Requirements

The input data set must contain either quarterly or monthly time series, and the data must be in chronological order. For the standard X-11 method, there must be at least three years of observations (12 for quarterly time series or 36 for monthly) in the input data sets or in each BY group in the input data set if a BY statement is used.

For the X-11-ARIMA method, there must be at least five years of observations (20 for quarterly time series or 60 for monthly) in the input data sets or in each BY group in the input data set if a BY statement is used.

Missing Values

Missing values at the beginning of a series to be adjusted are skipped. Processing starts with the first nonmissing value and continues until the end of the series or until another missing value is found.

Missing values are not allowed for the DATE= variable. The procedure terminates if missing values are found for this variable.

Missing values found in the PMFACTOR= variable are replaced by 100 for the multiplicative model (default) and by 0 for the additive model.

Missing values can occur in the output data set. If the time series specified in the OUTPUT statement is not computed by the procedure, the values of the corresponding variable are missing. If the time series specified in the OUTPUT statement is a moving average, the values of the corresponding variable are missing for the first *n* and last *n* observations, where *n* depends on the length of the moving average. Additionally, if the time series specified is an irregular component modified for extremes, only the modified values are given, and the remaining values are missing.

Prior Daily Weights and Trading-Day Regression

Suppose that a detailed examination of retail sales at ZXY Company indicates that certain days of the week have higher sales. In particular, Thursday, Friday and Saturday have approximately double the number of sales as Monday, Tuesday, and Wednesday, and no sales occur on Sunday. This means that months with five Saturdays would have higher sales than months with only four Saturdays.

This phenomenon is called a calendar effect; it can be handled in PROC X11 by using the PDWEIGHTS (Prior Daily WEIGHTS) statement or the TDREGR=option (Trading-Day REGRession). The PDWEIGHTS statement and the TDREGR=option can be used separately or together.

If the relative weights are known (as in the preceding) it is appropriate to use the PDWEIGHTS statement. If further residual calendar variation is present TDREGR=ADJUST should also be used. If you know that a calendar effect is present, but know nothing about the relative weights, use TDREGR=ADJUST without a PDWEIGHTS statement.

In this example, it is assumed that the calendar variation is due to both prior daily weights and residual variation. Thus both a PDWEIGHTS statement and TDREGR= ADJUST are specified.

Note that only the relative weights are needed; in the actual computations, PROC X11 normalizes the weights to sum to 7.0. If a day of the week is not present in the PDWEIGHTS statement, it is given a value of zero. Thus "sun=0" is not needed.

```
proc x11 data=sales;
   monthly date=date tdregr=adjust;
   var sales;
   tables a1 a4 b15 b16 C14 C15 c18 d11;
   pdweights mon=1 tue=1 wed=1 thu=2 fri=2 sat=2;
```

```
      output out=x11out a1=a1 a4=a4 b1=b1 c14=c14
                        c16=c16 c18=c18 d11=d11;
   run;
```

Tables of interest include A1, A4, B15, B16, C14, C15, C18, and D11. Table A4 contains the adjustment factors derived from the prior daily weights, table C14 contains the extreme irregular values excluded from trading-day regression, table C15 contains the trading day-regression results, table C16 contains the monthly factors derived from the trading-day regression, table C18 contains the final trading-day factors derived from the combined daily weights. Finally, table D11 contains the final seasonally adjusted series.

Adjustment for Prior Factors

Suppose now that a strike at ZXY Company during July and August of 1988 caused sales to decrease an estimated 50%. Since this is a one-time event with a known cause, it is appropriate to prior adjust the data to reflect the effects of the strike. This is done in PROC X11 through the use of PMFACTOR= *varname* (Prior Monthly FACTOR) on the MONTHLY statement.

In the following example, the PMFACTOR variable is named PMF. Since the estimate of the decrease in sales is 50%, PMF has a value of 50.0 for the observations corresponding to July and August, 1988, and a value of 100.0 for the remaining observations.

This prior adjustment to SALES is performed to SALES by computing (SALES/PMF) * 100.0. A value of 100.0 for PMF leaves SALES unchanged, while a value of 50.0 for PMF doubles SALES. This value is the estimate of what SALES would have been without the strike. The following example shows how this prior adjustment is accomplished.

```
   data sales; set sales;
      if '01jul1988'd <= date <= '01aug1988'd then pmf = 50;
      else pmf = 100;
   run;

   proc x11 data=sales;
      monthly date=date pmfactor=pmf;
      var sales;
      tables a1 a2 a3 d11;
      output out=x11out a1=a1 a2=a2 a3=a3 d11=d11;
   run;
```

Table A2 contains the prior monthly factors (the values of PMF), and Table A3 contains the prior adjusted series.

The YRAHEADOUT Option

For monthly data, the YRAHEADOUT option affects only tables C16 (regression trading-day adjustment factors), C18 (trading-day factors from combined daily weights), and D10 (seasonal factors). For quarterly data, only D10 is affected. Variables for all other tables have missing values for the forecast observations. The forecast values for a table are included only if that table is specified in the OUTPUT statement.

Tables C16 and C18 are calendar effects that are extrapolated by calendar composition. These factors are independent of the data once trading-day weights have been calculated. Table D10 is extrapolated by a linear combination of past values. If N is the total number of nonmissing observations for the analysis variable, this linear combination is given by

$$D10_t = \frac{1}{2}(3 \times D10_{t-12} - D10_{t-24}), \quad t = N+1, .., N+12$$

If the input data are monthly time series, 12 extra observations are added to the end of the output data set. (If a BY statement is used, 12 extra observations are added to the end of each BY group.) If the input data is a quarterly time series, four extra observations are added to the end of the output data set. (If a BY statement is used, four extra observations are added to each BY group.)

The DATE= variable (or _DATE_) is extrapolated for the extra observations generated by the YRAHEADOUT option, while all other ID variables will have missing values.

If ARIMA processing is requested, and if both the OUTEXTRAP and YRAHEADOUT options are specified in the PROC X11 statement, an additional 12 (4) observations are added to the end of output data set for monthly (quarterly) data after the ARIMA forecasts, using the same linear combination of past values as before.

Effect of Backcast and Forecast Length

Based on a number of empirical studies, (Dagum 1982a, 1982b, 1982c, Dagum and Laniel, 1987) one year of forecasts minimized revisions when new data become available. Two and three years of forecasts showed only small gains.

Backcasting improves seasonal adjustment but introduces permanent revisions at the beginning of the series and also at the end for series of length 8, 9 or 10 years. For series shorter than 7 years, the advantages of backcasting outweigh the disadvantages (Dagum, 1988).

Other studies (Pierce, 1980, Bobbit and Otto, 1990, Buszuwski, 1987) suggest "full forecasting"; that is, using enough forecasts to allow symmetric weights for the seasonal moving averages for the most current data. For example, if a 3x9 seasonal moving average was specified for one or more months using the MACURVES statement, five years of forecasts would be required. This is because the seasonal moving averages are performed on calendar months separately, and the 3x9 is an eleven-term centered moving average, requiring five observations before and after the current observation. Thus

```
   macurves dec='3x9';
```

would require five additional December values to compute the seasonal moving average.

Details of Model Selection

If an ARIMA statement is present, but no MODEL= is given, PROC X11 estimates and forecasts five predefined models and selects the best. This section describes the details of the selection criteria and the selection process.

The five predefined models used by PROC X11 are the same as those used by X11ARIMA/88 from Statistics Canada. These particular models, shown in Table 31.1 were chosen on the basis of testing a large number of economics series (Dagum, 1988) and should provide reasonable forecasts for most economic series.

Table 31.1. Five Predefined Models

Model #	Specification	Multiplicative	Additive
1	(0,1,1)(0,1,1)s	log transform	no transform
2	(0,1,2)(0,1,1)s	log transform	no transform
3	(2,1,0)(0,1,1)s	log transform	no transform
4	(0,2,2)(0,1,1)s	log transform	no transform
5	(2,1,2)(0,1,1)s	no transform	no transform

The selection process proceeds as follows. The five models are estimated and one-step-ahead forecasts are produced in the order shown in Table 31.1. As each model is estimated the following three criteria are checked:

- The Mean Absolute Percent Error (MAPE) for the last three years of the series must be less than 15 %.
- The significance probability for the Box-Ljung Chi-square for up to lag 24 for monthly (8 for quarterly) must greater than 0.05.
- The over-differencing criteria must not exceed 0.9.

The description of these three criteria are given in "Criteria Details." The default values for these criteria are those used by X11ARIMA/88 from Statistics Canada; these defaults can be changed by the MAPECR=, CHICR= and OVDIFCR= options.

A model that fails any one of these three criteria is excluded from further consideration. In addition, if the ARIMA estimation fails for a given model, a warning is issued, and the model is excluded. The final set of all models considered are those that pass all three criteria and are estimated successfully. From this set, the model with the smallest MAPE for the last three years is chosen.

If all five models fail, ARIMA processing is skipped for the variable being processed, and the standard X-11 seasonal adjustment is performed. A note is written to the log with this information.

The chosen model is then used to forecast the series one or more years (determined by the FORECAST= option on the ARIMA statement). These forecasts are appended on the original data (or the prior and calendar-adjusted data).

If a BACKCAST= is specified, the chosen model form is used, but the parameters are reestimated using the reversed series. Using these parameters, the reversed series is forecasted for the number of years specified by the BACKCAST= option. These forecasts are then reversed and appended to the beginning of the original series, or the prior and calendar-adjusted series, to produce the backcasts.

Note that the final selection rule (the smallest MAPE using the last three years) emphasizes the quality of the forecasts at the end of the series. This is consistent with the purpose of the X-11-ARIMA methodology, namely, to improve the estimates of seasonal factors and thus minimize revisions to recent past data as new data become available.

Criteria Details

The Mean Absolute Percent Error (MAPE)

For the MAPE criteria testing, only the last three years of the original series (or prior and calendar adjusted series) is used in computing the MAPE.

Let y_t, *t*=1,..,*n* be the last three years of the series, and denote its one-step-ahead forecast by $\hat{y}_t$, where n=36 for a monthly series, and n=12 for a quarterly series.

With this notation, the MAPE criteria is computed as

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{|y_t|}$$

Box-Ljung Chi-Square

The Box-Ljung Chi-Square is a lack of fit test using the model residuals. This test statistic is computed using the Ljung-Box formula

$$\chi_m^2 = n(n+2) \sum_{k=1}^{m} \frac{r_k^2}{(n-k)}$$

where *n* is the number of residuals that can be computed for the time series, and

$$r_k = \frac{\sum_{t=1}^{n-k} a_t a_{t+k}}{\sum_{t=1}^{n} a_t^2}$$

where the a_t's are the residual sequence. This formula has been suggested by Ljung and Box as yielding a better fit to the asymptotic chi-square distribution. Some simulation studies of the finite sample properties of this statistic are given by Davies, Triggs, and Newbold (1977) and by Ljung and Box (1978).

For monthly series, m=24, while for quarterly series, m=8.

Over-Differencing Test

From Table 31.1 you can se that all models have a single seasonal MA factor and at most two nonseasonal MA factors. Also, all models have seasonal and nonseasonal differencing. Consider model 2 applied to a monthly series y_t with $E(y_t) = \mu$:

$$(1 - B^1)(1 - B^{12})(y_t - \mu) = (1 - \theta_1 B - \theta_2 B^2)(1 - \theta_3 B^{12})a_t$$

If $\theta_3 = 1.0$, then the factors $(1 - \theta_3 B^{12})$ and $(1 - B^{12})$ will cancel, resulting in a lower-order model.

Similarly, if $\theta_1 + \theta_2 = 1.0$,

$$(1 - \theta_1 B - \theta_2 B^2) = (1 - B)(1 - \alpha B)$$

for some $\alpha \neq 0.0$. Again, this results in cancelation and a lower order model.

Since the parameters are not exact, it is not reasonable to require that

$$\theta_3 < 1.0 \text{ and } \theta_1 + \theta_2 < 1.0$$

Instead, an approximate test is performed by requiring that

$$\theta_3 \leq 0.9 \text{ and } \theta_1 + \theta_2 \leq 0.9$$

The default value of 0.9 can be changed by the OVDIFCR= option. Similar reasoning applies to the other models.

ARIMA Statement Options for the Five Predefined Models

The following table lists the five predefined models and gives the equivalent MODEL= parameters in a PROC X11 ARIMA statement.

In all models except the fifth, a log transformation is performed before the ARIMA estimation for the multiplicative case; no transformation is performed for the additive case. For the fifth model, no transformation is done for either case.

The multiplicative case is assumed in the following table. The indicated seasonality *s* in the specification is either 12 (monthly), or 4 (quarterly). The MODEL statement assumes a monthly series.

Table 31.2. ARIMA Statements Options for Predefined Models

Model	ARIMA Statement Options
(0,1,1)(0,1,1)s	MODEL=(Q=1 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(0,1,2)(0,1,1)s	MODEL=(Q=2 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(2,1,0)(0,1,1)s	MODEL=(P=2 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(0,2,2)(0,1,1)s	MODEL=(Q=2 SQ=1 DIF=2 SDIF=1) TRANSFORM=LOG
(2,1,2)(0,1,1)s	MODEL=(P=2 Q=2 SQ=1 DIF=1 SDIF=1)

OUT= Data Set

The OUT= data set specified in the OUTPUT statement contains the BY variables, if any; the ID variables, if any; and the DATE= variable if the DATE= option is given, or _DATE_ if the DATE= option is not specified.

In addition, the variables specified by the option

tablename =var1 var2 . . . varn

are placed in the OUT= data set. A list of tables available for monthly and quarterly series is given in Table 31.3.

The OUTSPAN Data Set

OUTSPAN= *SAS-data-set*

This option is specified on the PROC statement, and writes the sliding spans results to the specified output data set. The OUTSPAN data set contains the following variables.

- A1, a numeric variable that is a copy of the original series truncated to the current span. Note that overlapping spans will contain identical values for this variable.
- C18, a numeric variable that contains the Trading Day Factors for the seasonal adjustment for the current span.
- D10, a numeric variable that contains the Seasonal Factors for the seasonal adjustment for the current span.
- D11, a numeric variable that contains the Seasonally Adjusted Series for the current span.
- DATE, a numeric variable that contains the date within the current span.
- SPAN, a numeric variable that contains the current span. The first span is the earliest span, i.e., the one with the earliest begin date.
- VARNAME, a character variable containing the name of each variable in the VAR list. A separate sliding spans analysis is performed on each variable in the VAR list.

OUTSTB= Data Set

The output data set produced by the OUTSTB= option of the PROC X11 statement contains the information in the analysis of variance on table D8 (Final Unmodified

S-I Ratios). This analysis of variance, following table D8 in the printed output, tests for stable seasonality (refer to U.S. Bureau of the Census, 1967, Appendix A). The variables in this data are:

- VARNAME, a character variable containing the name of each variable in the VAR list.
- TABLE, a character variable specifying the table from which the analysis of variance is performed. When ARIMA processing is requested, and two passes of X11 are required (when TDREGR=PRINT, TEST, or ADJUST), Table D8 and the stable seasonality test are computed twice; once in the initial pass, then again in the final pass. Both of these computations are put in the OUTSTB data set and are identified by D18.1 or D18.2 respectively.
- SOURCE, a character variable corresponding to the "source" column in the Analysis of Variance table following Table D8.
- SS, a numeric variable containing the sum of squares associated with the corresponding source term.
- DF, a numeric variable containing the degrees of freedom associated with the corresponding source term.
- MS, a numeric variable containing the mean square associated with the corresponding source term. MS is missing for the source term "Total."
- F, a numeric variable containing the F statistic for the "Between" source term. F will be missing for all other source terms.
- PROBF, a numeric variable containing the significance level for the F statistic. PROBF is missing for the source term "Total" and "Error."

OUTTDR= Data Set

The trading-day regression results (tables B15 and C15) are written to the OUTTDR= data set, which contains the following variables:

- VARNAME, a character variable containing the name of the VAR variable being processed.
- TABLE, a character variable containing the name of the table. It can only have values B15 (Preliminary Trading-Day Regression) or C15 (Final Trading-Day Regression).
- _TYPE_, a character variable whose value distinguishes the three distinct table format types. These types are (a) the regression, (b) the listing of the standard error associated with length-of-month, and (c) the Analysis of Variance. The first seven observations in the OUTTDR data set correspond to the regression on days of the week, thus the _TYPE_ variable is given the value "REGRESS" (day-of-week regression coefficient). The next four observations correspond to 31, 30, 29, and 28 day months and are given the value _TYPE_ = LOM_STD (length-of-month standard errors). Finally the last three observations correspond to the Analysis of Variance table, and _TYPE_ = ANOVA.

- PARM, a character variable, further identifying the nature of the observation. PARM is set to blank for the three _TYPE_ = ANOVA observations.
- SOURCE, a character variable containing the source in the regression. This variable is missing for all _TYPE_ = REGRESS and LOM_STD.
- CWGT, a numeric variable containing the combined trading-day weight (prior weight + weight found from regression). The variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA .
- PRWGT, a numeric variable containing the prior weight. The prior weight is 1.0 if PDWEIGHTS are not specified. This variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA .
- COEFF, a numeric variable containing the calculated regression coefficient for the given day. This variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA .
- STDERR, a numeric variable containing the standard errors. For observations with _TYPE_ = REGRESS, this is the standard error corresponding to the regression coefficient. For observations with _TYPE_ = LOM_STD, this is standard error for the corresponding length-of-month. This variable is missing for all _TYPE_ = ANOVA .
- T1, a numeric variable containing the *t*-statistic corresponding to the test that the combined weight is different from the prior weight. This variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA .
- T2, a numeric variable containing the *t*-statistic corresponding to the test that the combined weight is different from 1.0 . This variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA.
- PROBT1, a numeric variable containing the significance level for *t*-statistic T1. The variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA.
- PROBT2, a numeric variable containing the significance level for *t*-statistic T2. The variable is missing for all _TYPE_ = LOM_STD and _TYPE_ = ANOVA .
- SS, a numeric variable containing the sum of squares associated with the corresponding source term. This variable is missing for all _TYPE_ = REGRESS and LOM_STD.
- DF, a numeric variable containing the degrees of freedom associated with the corresponding source term. This variable is missing for all _TYPE_ = REGRESS and LOM_STD.
- MS, a numeric variable containing the mean square associated with the corresponding source term. This variable is missing for the source term Total and for all _TYPE_ = REGRESS and LOM_STD.
- F, a numeric variable containing the F statistic for the Regression source term. The variable is missing for the source terms Total and Error, and for all _TYPE_ = REGRESS and LOM_STD.

- PROBF, a numeric variable containing the significance level for the F statistic. This variable is missing for the source term Total and Error and for all _TYPE_ = REGRESS and LOM_STD.

Printed Output

The output from PROC X11, both printed tables and the series written to the OUT= data set, depends on whether the data is monthly or quarterly. For the printed tables, the output depends further on the value of the PRINTOUT= option and the TABLE statement, along with other options specified.

The printed output is organized into tables identified by a part letter and a sequence number within the part. The seven major parts of the X11 procedure are as follows.

A	prior adjustments (optional)
B	preliminary estimates of irregular component weights and regression trading-day factors
C	final estimates of irregular component weights and regression trading-day factors
D	final estimates of seasonal, trend cycle, and irregular components
E	analytical tables
F	summary measures
G	charts

Table 31.3 describes the individual tables and charts. Most tables apply both to quarterly and monthly series. Those that apply only to a monthly time series are indicated by an "M" in the notes section, while "P" indicates the table is not a time series, and is only printed, not output to the OUT= data set.

Table 31.3. Table Names and Descriptions

Table	Description	Notes
A1	original series	M
A2	prior monthly adjustment factors	M
A3	original series adjusted for prior monthly factors	M
A4	prior trading-day adjustments	M
A5	prior adjusted or original series	M
A13	ARIMA forecasts	
A14	ARIMA backcasts	
A15	prior adjusted or original series extended by arima backcasts, forecasts	
B1	prior adjusted or original series	
B2	trend cycle	
B3	unmodified seasonal-irregular (S-I) ratios	
B4	replacement values for extreme S-I ratios	
B5	seasonal factors	

Table 31.3. (continued)

Table	Description	Notes
B6	seasonally adjusted series	
B7	trend cycle	
B8	unmodified S-I ratios	
B9	replacement values for extreme S-I ratios	
B10	seasonal factors	
B11	seasonally adjusted series	
B13	irregular series	
B14	extreme irregular values excluded from trading-day regression	M
B15	preliminary trading-day regression	M,P
B16	trading-day adjustment factors	M
B17	preliminary weights for irregular components	
B18	trading-day factors derived from combined daily weights	M
B19	original series adjusted for trading-day and prior variation	M
C1	original series modified by preliminary weights and adjusted for trading-day and prior variation	
C2	trend cycle	
C4	modified S-I ratios	
C5	seasonal factors	
C6	seasonally adjusted series	
C7	trend cycle	
C9	modified S-I ratios	
C10	seasonal factors	
C11	seasonally adjusted series	
C13	irregular series	
C14	extreme irregular values excluded from trading-day regression	M
C15	final trading-day regression	M,P
C16	final trading-day adjustment factors derived from regression coefficients	M
C17	final weight for irregular components	
C18	final trading-day factors derived from combined daily weights	M
C19	original series adjusted for trading-day and prior variation	M
D1	original series modified for final weights and adjusted for trading-day and prior variation	
D2	trend cycle	
D4	modified S-I ratios	
D5	seasonal factors	
D6	seasonally adjusted series	
D7	trend cycle	
D8	final unmodified S-I ratios	
D9	final replacement values for extreme S-I ratios	
D10	final seasonal factors	
D11	final seasonally adjusted series	
D12	final trend cycle	
D13	final irregular series	
E1	original series with outliers replaced	

Table 31.3. (continued)

Table	Description	Notes
E2	modified seasonally adjusted series	
E3	modified irregular series	
E4	ratios of annual totals	P
E5	percent changes in original series	
E6	percent changes in final seasonally adjusted series	
F1	MCD moving average	
F2	summary measures	P
G1	chart of final seasonally adjusted series and trend cycle	P
G2	chart of S-I ratios with extremes, S-I ratios without extremes, and final seasonal factors	P
G3	chart of S-I ratios with extremes, S-I ratios without extremes, and final seasonal factors in calendar order	P
G4	chart of final irregular and final modified irregular series	P

The PRINTOUT= Option

The PRINTOUT= option controls printing for groups of tables. See the "TABLES Statement" in this chapter for details on specifying individual tables. The following list gives the tables printed for each value of the PRINTOUT= option.

STANDARD (26 tables)	A1-A4, B1, C13-C19, D8-D13, E1-E6, F1, F2.
LONG (40 tables)	A1-A5, A13-A15, B1, B2, B7, B10, B13-B15, C1, C7, C10, C13-C19, D1, D7-D11, D13, E1-E6, F1, F2.
FULL (62 tables)	A1-A5, A13-A15, B1-B11, B13-B19, C1-C11, C13-C19, D1, D2, D4-D12, E1-E6, F1, F2.

The actual number of tables printed depends on the options and statements specified. If a table is not computed, it is not printed. For example, if TDREGR=NONE is specified, none of the tables associated with the trading-day are printed.

The CHARTS= Option

Of the four charts listed in Table 31.3, G1 and G2 are printed by default (CHARTS=STANDARD). Charts G3 and G4 are printed when CHARTS=FULL is specified. See the "TABLES Statement" later in this chapter for details in specifying individual charts.

Stable, Moving and Combined Seasonality Tests on the Final Unmodified SI Ratios (Table D8)

Past releases of PROC X11 printed the "Stable Seasonality Test" after Table D8. Two additional tests have been added and are printed the just after the "Stable Seasonality Test". The motivation, interpretation, and statistical details of all these tests are now given.

Motivation

The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated constantly (stable) or in an evolving fashion from year to year (moving seasonality).

To determine if stable seasonality if present in a series, PROC X11 computes a one-way analysis of variance using the seasons (months or quarters) as the factor on the Final Unmodified SI Ratios (Table D8). This is the appropriate table to use since the removal of the trend-cycle is equivalent to detrending. PROC X11 prints this test, labeled "Stable Seasonality Test" immediately after the Table D8. This test has not changed from previous releases.

The X11 seasonal adjustment method allows for slowing evolving seasonality. PROC X11 now computes and prints a test for seasonality when it is evolving or moving. The test is a two-way analysis of variance using months (or quarters) and years. As in the "Stable Seasonality Test", this analysis of variance is performed on the Final Unmodified SI Ratios (Table D8). PROC X11 prints this test, labeled "Moving Seasonality Test" after the "Stable Seasonality Test".

The final new test that PROC X11 computes is a combined or joint test of both stable and moving seasonality. This test combines the two F-tests previously described, along with the Kruskal-Wallis Chi-squared test for the stable seasonality to determine "identifiable" seasonality. This test, labeled "Combined Test for the Presence of Identifiable Seasonality", is printed after the "Moving Seasonality Test".

Interpretation and Statistical Details

The "Stable Seasonality Test" is a one-way analysis of variance on the "Final Unmodified SI Ratios" with seasons (months or quarters) as the factor.

To determine if stable seasonality if present in a series, PROC X11 computes a one-way analysis of variance using the seasons (months or quarters) as the factor on the Final Unmodified SI Ratios (Table D8). This is the appropriate table to use since the removal of the similar to detrending.

A large F and small significance level is evidence that a significant amount of variation in the SI-ratios is due to months or quarters, which in turn is evidence of seasonality; the null hypothesis of no month/quarter effect is rejected.

Conversely, a small F and large significance level (close to 1.0) is evidence that variation due to month or quarter could be due random error and the null hypothesis of no month/quarter effect is not rejected. The interpretation and utility of seasonal adjustment is problematical under such conditions.

The F-test for moving seasonality is performed by a two-way analysis of variance. The two factors are seasons (months or quarters) and years. The years effect is tested separately; the null hypothesis is no effect due to years after accounting for variation due to months or quarters.)

The significance level reported in both the moving and stable seasonality test is only approximate. Table D8, the Final Unmodified SI Ratios is constructed from an averaging operation which induces a correlation in the residuals from which which the F-test is computed. Hence the computed F-statistic differs from an exact F; see Cleveland and Devlin, 1980 for details.

The test for identifiable seasonality is performed by combining the F-tests for stable and moving seasonality, along with a Kruskal-Wallis test for stable seasonality. The description below is based on Dagum, (1980); for further details, see Lothian and Morry, 1978b.

Let F_s and F_m denote the F-value for the stable and moving seasonality tests respectively. The combined test is performed as follows.

1) If the null hypothesis in the moving seasonality tests is not rejected at the 0.10 % level (one thousandths percent), the seasonality is not identifiable.

2) If the null hypothesis in 1) is rejected, but the moving seasonality null hypothesis is not rejected at the 5.0% level, then compute the following quantities:

$$T_1 = \frac{7}{F_m - F_s},$$

$$T_2 = \frac{3F_m}{F_s}.$$

Let T denote the simple average of T_1 and T_2:

$$T = \frac{(T_1 + T_2)}{2};$$

if $T \geq 1.0$, the null hypothesis of identifiable seasonality *not* present is accepted.

3) If the moving seasonality f-test based on F_M passes, but one of the two statistics based on the T's fails, or the Kruskal-Wallis Chi-squared test fails at the 1% level, the then PROC X11 prints "Identifiable Seasonality Probably Present"

4) If the F_S, F_M and the Kruskal-Wallis Chi-squared test pass, then the null hypothesis (of identifiable seasonality *not* present if rejected, and PROC X11 prints "Identifiable Seasonality Present".

Tables Written to the OUT= data set

All tables that are time series can be written to the OUT= data set. However, depending on the specified options and statements, not all tables are computed. When a table is not computed, but is requested in the OUTPUT statement, the resulting variable has all missing values.

For example, if the PMFACTOR= option is not specified, table A2 is not computed, and requesting this table in the OUTPUT statement results in the corresponding variable having all missing values.

The trading-day regression results, tables B15 and C15, although not written to the OUT= data set, can be written to an output data set; see the "OUTTDR=" option for details.

Printed Output Generated by Sliding Spans Analysis

Table S 0.A

Table S 0.A gives the variable name, the length and number of spans, and the beginning and ending dates of each span.

Table S 0.B

Table S 0.B gives the summary of the two f-tests performed during the standard X11 seasonal adjustments for stable and moving seasonality on table D8, the final SI ratios. These tests are described in "Printed Output" in the "PROC X11" chapter.

Table S 1.A

Table S 1.A gives the range analysis of seasonal factors. This includes the means for each month (or quarter) within a span, the maximum percent difference across spans for each month and the average. The minimum and maximum within a span is also indicated.

For example, for a monthly series and an analysis with four spans, the January row would contain a column for each span, with the value representing the average seasonal factor (Table D10) over all January calendar months occurring within the span. Beside each span column is a character column with either a MIN, MAX or blank value, indicating which calendar month had the minimum and maximum value over that span.

Denote the average over the j-th calendar month in span k, k=1,..,4 by $\bar{S}_j$(k); then the maximum percent difference (MPD) for month j is defined by

$$MPD_j = \frac{max_{k=1,..,4}\bar{S}_j(k) - min_{k=1,..,4}\bar{S}_j(k)}{min_{k=1,..,4}\bar{S}_j(k)}$$

The last numeric column of Table S 1.A is the average value over all spans for each calendar month, with the minimum and maximum row flagged as in the span columns.

Table S 1.B

Table S 1.B gives a summary of range measures for each span. The first column, Range Means, is calculated by computing the maximum and minimum over all months or quarters in a span, the taking the difference. The next column is the range ratio means, which is simply the ratio of the previously described maximum and minimum. The next two columns are the minimum and maximum seasonal factors over the entire span, while the range sf column is the difference of these. Finally, the last column is the ratio of the Max SF and Min SF columns.

Breakdown Tables

Table S 2.A.1 begins the breakdown analysis for the various series considered in the sliding spans analysis. The key concept here is the MPD described in the Introduction and in "Computational Details" above. For a month or quarter that appears in two or more spans, the maximum percent difference is computed and tested against a cutoff level. If it exceeds this cutoff, it is counted as an instance of exceeding the level. It is of interest to see if such instances fall disproportionately in certain months and years. Tables S 2.A.1 - S 6.A.3 display this breakdown for all series considered.

Table S 2.A.1

Table S 2.A.1 gives the monthly (quarterly) breakdown for the seasonal factors (table D10). The first column identifies the month or quarter. The next column is the number of times the MPD for D10 exceeded 3.0%, followed by the total count. The last is the average maximum percentage difference for the corresponding month or quarter.

Table S 2.A.2

Table S 2.A.2 gives the same information as Table S 2.A.1, but on a yearly basis.

Table S 2.A.3

The description of Table S 2.A.3 requires the definition of "Sign Change" and "Turning Point".

First, some motivation. Recall that for a highly stable series, adding or deleting a small number of observations should not affect the estimation of the various components of a seasonal adjustment procedure.

Consider Table D10, the seasonal factors in a sliding spans analysis that uses 4 spans. For a given observation t, looking across the 4 spans, we can easily pick out large differences if they occur. More subtle differences can occur when estimates go from above to below (or vice versa) a base level. In the case of multiplicative model, the seasonal factors have a base level of 100.0. So it is useful to enumerate those instances where both a large change occurs (an MPD value exceeding 3.0%) and a change of sign (with respect to the base) occur.

Let B denote the base value (which in general depends on the component being considered and the model type, multiplicative or additive). If, for span 1, $S_t(1)$ is below B (i.e., $S_t(1)$-B is negative) and for some subsequent span k, $S_t(k)$ is above B (i.e., $S_t(k)$-B is positive), then an positive "Change in Sign" has occurred at observation t. Similarly, if, for span 1, $S_t(1)$ is above B, and for some subsequent span k, $S_t(k)$ is below B, then a negative "Change in Sign" has occurred. Both cases, positive or negative, constitute a "Change in Sign"; the actual direction indicated in tables S 7.A-S 7.E, which will be described below.

Another behavior of interest occurs when component estimates increase then decrease (or vice versa) across spans for a given observation. Using the example above, the seasonal factors at observation t could first increase, then decrease across the 4 spans.

This behavior, combined with an MPD exceeding the level is of interest in questions of stability.

Again, consider Table D10, the seasonal factors in a sliding spans analysis that uses 4 spans. For a given observation t, (containing at least three spans), note the level of D10 for the first span. Continue across the spans until a difference of 1.0% or greater occurs (or no more spans are left), noting whether the difference is up or down. If the difference is up, continue until a difference of 1.0% or greater occurs downward (or no more spans are left). If such an up-down combination occurs, the observation is counted as an up-down turning point. A similar description occurs for a down-up turning point. Tables S 7.A-S 7.E, described below, show the occurrence of turning points, indicating whether up-down or down-up. Note that it requires at least three spans to test for a turning point. Hence Tables S 2.A.3 - S 6.A.3 show a reduced number in the "Turning Point" row for the "Total Tested" column, and in Tables S 7.A - S 7.E, the turning points symbols can only occur where three or more spans overlap.

With these descriptions of sign change and turning point, we now describe Table S 2.A.3. The first column gives the type or category, the second gives the total number of observations falling into the category, the third column gives the total number tested, and the last column gives the percentage for the number found in the category.

The first category (row) of the table is for flagged observations, i.e., those observations where the MPD exceeded the appropriate cutoff level (3.0% is default for the seasonal factors.) The second category is for level changes, while the third category is for turning points. The fourth category is for flagged sign changes, i.e., for those observations that are sign changes, how many are also flagged. Note the total tested column for this category equals the number found for sign change, reflecting the definition of the fourth category.

The fifth column is for flagged turning points, i.e., for those observations that are turning points, how many are also flagged.

The footnote to Table S 2.A.3 gives the Census Bureau recommendation for thresholds, as described in "Computational Details" earlier in this section.

Table S 2.B

Table S 2.B gives the histogram of flagged for seasonal factors (Table D10) using the appropriate cutoff value (default 3.0%). This table looks at the spread of the number of times the MPD exceeded the corresponding level. The range is divided up into four intervals: 3.0%-4.0%, 4.0%-5.0%, 5.0%-6.0% and greater than 6.0%. The first column shows the symbol used in table S 7.A; the second column gives the range in interval notation, and the last column gives the number found in the corresponding interval. Note that the sum of the last column should agree with the "Number Found" column of the "Flagged MPD" row in Table S 2.A.3.

Table S 2.C

Table S 2.C gives selected percentiles for the MPD for the seasonal factors (Table D10).

Tables S 3.A.1 - S 3.A.3

These table relate to the Trading Day Factors (Table C18), and follow the same format as Tables S 2.A.1-S 2.A.3. The only difference between these tables and S 2.A.1-S 2.A.3 is the default cutoff value of 2.0% instead of the 3.0% used for the Seasonal Factors.

Tables S 3.B, S 3.C

These tables, applied to the Trading Day Factors (Table C18), are the same format as tables S 2.B - S 2.C. The default cutoff value is different, with corresponding differences in the intervals in S 3.B.

Tables S 4.A.1 - S 4.A.3

These table relate to the Seasonally Adjusted Series (Table D11), and follow the same format as Tables S 2.A.1-S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 4.B, S 4.C

These tables, applied to the Seasonally Adjusted Series (Table D11) are the same format as tables S 2.B - S 2.C.

Tables S 5.A.1 - S 5.A.3

These table relate to the Month-to-Month (or Quarterly-to-Quarterly) differences in the Seasonally Adjusted Series, and follow the same format as Tables S 2.A.1-S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 5.B, S 5.C

These tables, applied to the Month-to-Month (or Quarterly-to-Quarterly) differences in the Seasonally Adjusted Series, are the same format as tables S 2.B - S 2.C. The same default cutoff value of 3.0% is used.

Tables S 6.A.1 - S 6.A.3

These table relate to the Year-to-Year differences in the Seasonally Adjusted Series, and follow the same format as Tables S 2.A.1-S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 6.B, S 6.C

These tables, applied to the Year-to-Year differences in the Seasonally Adjusted Series, are the same format as tables S 2.B - S 2.C. The same default cutoff value of 3.0% is used.

Table S 7.A

Table S 7.A gives the entire listing of the Seasonal Factors (Table D10) for each span. The first column gives the date for each observation included in the spans. Note that the dates do not cover the entire original data set. Only those observations included in one or more spans are listed.

The next N columns (where N is the number of spans) are the individual spans starting at the earliest span. The span columns are labeled by their beginning and ending dates.

Following the last span is the "Sign Change" column. As explained in the description of Table S 2.A.3, a sign change occurs at a given observation when the seasonal factor estimates go from above to below, or below to above, a base level. For the seasonal factors, 100.0 is the base level for the multiplicative model, 0.0 for the additive model. A blank value indicates no sign change, a "U" indicates a movement "upwards" from the base level and a "D" indicates a movement "downwards" from the base level.

The next column is the "Turning Point" column. As explained in the description of Table S 2.A.3, a turning point occurs when there is an upward then downward movement, or downward then upward movement of sufficient magnitude. A blank value indicates no turning point, a "U-D" indicates a movement "upwards then downwards" and a "D-U" indicates a movement "downwards then upwards".

The next column is the maximum percent difference (MPD). This quantity, described in "Computational Details" above, is the main computation for sliding spans analysis. A measure of how extreme the MPD value is given in the last column, the "Level of Excess" column. The symbols used and their meaning is described in Table S 2.A.3. If a given observation has exceeded the cutoff, the level of excess column is blank.

Table S 7.B

Table S 7.B gives the entire listing of the Trading Day Factors (Table C18) for each span. The format of this table is exactly like Table S 7.A.

Table S 7.C

Table S 7.C gives the entire listing of the Seasonally Adjusted Data (Table D11) for each span. The format of this table is exactly like Table S 7.A except for the "Sign Change" column, which is not printed. The Seasonally Adjusted Data has the same units as the original data; there is no natural base level as in the case of a percentage. Hence the sign change is not appropriate for D11.

Table S 7.D

Table S 7.D gives the entire listing of the Month-to-Month (or Quarter-to-Quarter) Changes in Seasonally Adjusted Data for each span. The format of this table is exactly like Table S 7.A.

Table S 7.E

Table S 7.E gives the entire listing of the Year-to-Year Changes in Seasonally Adjusted Data for each span. The format of this table is exactly like Table S 7.A.

Printed Output from the ARIMA Statement

The information printed by default for the ARIMA model includes the parameter estimates, their approximate standard errors, t ratios, and variances, the standard deviation of the error term, and the AIC and SBC statistics for the model. In addition, a criteria summary for the chosen model is given that shows the values for each of the three test criteria and the corresponding critical values.

If the PRINTALL option is specified, a summary of the Nonlinear Estimation Optimization and a table of Box-Ljung Statistics is also produced. If the automatic model selection is used, this information is printed for each of the five predefined models. Lastly, a Model Selection Summary is printed, showing the final model chosen.

ODS Table Names

PROC X11 assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. For more information on ODS, see Chapter 8, “Using the Output Delivery System.”

Note: For monthly and quarterly tables use the ODSNAME MonthlyTables and QuarterlyTables; For brevity, only the MonthlyTables are listed here; the QuarterlyTables are simply duplicates. Printing of individual tables can be specified by using the TABLES table_name which is not listed here. Printing groups of tables is specified in the MONTHLY and QUARTERLY statements by specifying the option PRINTOUT=NONE|STANDARD|LONG|FULL. The default is PRINTOUT=STANDARD.

Table 31.4. ODS Tables Produced in PROC X11

ODS Table Name	Description	Option
ODS Tables Created by the MONTHLY and QUARTERLY Statements		
Preface	X11 Seasonal Adjustment Program Information giving credits, dates, etc.	Always printed unless NOPRINT
A1	Table A1: OriginalSeries	
A2	Table A2: Prior Monthly	
A3	Table A3: Original Series Adjusted for Prior Monthly Factors	
A4	Table A4: Prior Trading Day Adjustment Factors With and Without Length of Month Adjustment	
A5	Table A5: Original Series Adjusted for Priors	
B1	Table B1: Original Series or Original Series Adjusted for Priors	
B2	Table B2: Trend Cycle - Centered nn-Term Moving Average	
B3	Table B3: Unmodified SI Ratios	
B4	Table B4: Replacement Values for Extreme SI Ratios	
B5	Table B5: Seasonal Factors	
B6	Table B6: Seasonally Adjusted Series	
B7	Table B7: Trend Cycle - Henderson Curve	
B8	Table B8: Unmodified SI Ratios	
B9	Table B9: Replacement Values for Extreme SI Ratios	
B10	Table B10: Seasonal Factors	
B11	Table B11: Seasonally Adjusted Series	
B13	Table B13: Irregular Series	

Table 31.4. (continued)

ODS Table Name	Description	Option
B15	Table B15: Preliminary Trading Day Regression	
B16	Table B16: Trading Day Adjustment Factors Derived from Regression	
B17	Table B17: Preliminary Weights for Irregular Component	
B18	Table B18: Trading Day Adjustment Factors from Combined Weights	
B19	Table B19: Original Series Adjusted for Preliminary Comb. TD. Wgts.	
C1	Table C1: Original Series Adjusted for Preliminary Weights	
C2	Table C2: Trend Cycle - Centered nn-Term Moving Average	
C4	Table C4: Modified SI Ratios	
C5	Table C5: Seasonal Factors	
C6	Table C6: Seasonally Adjusted Series	
C7	Table C7 Trend Cycle - Henderson Curve	
C9	Table C9: Modified SI Ratios	
C10	Table C10: Seasonal Factors	
C11	Table C11: Seasonally Adjusted Series	
C13	Table C13: Irregular Series	
C15	Table C15: Final Trading Day Regression	
C16	Table C16: Trading Day Adjustment Factors Derived from Regression	
C17	Table C17: Final Weights for Irregular Component	
C18	Table C18: Trading Day Adjustment Factors from Combined Weights	
C19	Table C19: Original Series Adjusted for Final Comb. TD. Wgts.	
D1	Table D1: Original Series Adjusted for Final Weights nn-Term Moving Average	
D4	Table D4: Modified SI Ratios	
D5	Table D5: Seasonal Factors	
D6	Table D6: Seasonally Adjusted Series	
D7	Table D7: Trend Cycle - Henderson Curve	
D8	Table D8: Final Unmodified SI Ratios	
D10	Table D10: Final Seasonal Factors	
D11	Table D11: Final Seasonally Adjusted Series	
D12	Table D12: Final Trend Cycle - Henderson Curve	
D13	Table D13: Final Irregular Series	

Table 31.4. (continued)

ODS Table Name	Description	Option
E1	Table E1: Original Series Modified for Extremes	
E2	Table E2: Modified Seasonally Adjusted Series	
E3	Table E3: Modified Irregular Series	
E5	Table E5: Month-to-Month Changes in Original Series	
E6	Table E6: Month-to-Month Changes in Final Seasonally Adj. Series	
F1	Table F1: MCD Moving Average	
A13	Table A13: ARIMA Forecasts	ARIMA statement
A14	Table A14: ARIMA Backcasts	ARIMA statement
A15	Table A15: Arima Extrapolation	ARIMA statement
B14	Table B14: Irregular Values Excluded from Trading Day Regression	
C14	Table C14: Irregular Values Excluded from Trading Day Regression	
D9	Table D9: Final Replacement Values	
PriorDailyWgts	Adjusted Prior Daily Weights	
TDR_0	Final/ Preliminary Trading Day Regression, part 1	MONTHLY only, TDREGR=ADJUST, TEST
TDR_1	Final/ Preliminary Trading Day Regression, part 2	MONTHLY only, TDREGR=ADJUST, TEST
StandErrors	Standard Errors of Trading Day Adjustment Factors	MONTHLY only, TDREGR=ADJUST, TEST
D9A	Year to Year Change in Irregular and Seasonal Components And Moving Seasonality Ratio	
StableSeasTest	Stable Seasonality Test	MONTHLY only
StableSeasFTest	Stable Seasonality Test	MONTHLY only
f2a	F2 Summary Measures, part 1	
f2b	F2 Summary Measures, part 2	
f2c	F2 Summary Measures, part 3	
f2d	I/C Ratio for Month/Quarterly Span	

Table 31.4. (continued)

ODS Table Name	Description	Option
f2f	Avg % Change with regard to Sign and Std. Over Span	
E4	Differences or Ratios of Annual Totals, Original and Adjusted Series	
ChartG1	Chart G1	
ChartG2	Chart G2	
ODS Tables Created by the ARIMA Statement		
CriteriaSummary	Criteria Summary	ARIMA statement
ConvergeSummary	Convergence Summary	
ArimaEst	Arima estimation results, part 1	
ArimaEst2	Arima estimation results, part 2	
Model_Summary	Model Summary	
Ljung_BoxQ	Table of Ljung-Box Q Statistics	
A13	Table A13: ARIMA Forecasts	
A14	Table A14: ARIMA Backcasts	
A15	Table A15: Arima Extrapolation	
ODS Tables Created by the SSPAN Statement		
SPR0A_1	S 0.A Sliding Spans Analysis, Number, Length of Spans	default printing
SpanDates	S 0.A Sliding Spans Analysis: Dates of Spans	
SPR0B	S 0.B Summary of F-tests for Stable and Moving Seasonality	
SPR1_1	S 1.A Range Analysis of Seasonal Factors	
SPR1_b	S 1.B Summary of Range Measures	
SPRXA	2XA.1 Breakdown of Differences by Month or Qtr	
SPRXB_2	S X.B Histogram of Flagged Observations	
SPRXA_2	S X.A.2 Breakdown of Differences by Year	
MpdStats	S X.C: Statistics for Maximum Percentage Differences	
S_X_A_3	S 2.X.3 Breakdown Summary of Flagged Observations	
SPR7_X	S 7.X Sliding Spans Analysis	PRINTALL

Examples

Example 31.1. Component Estimation - Monthly Data

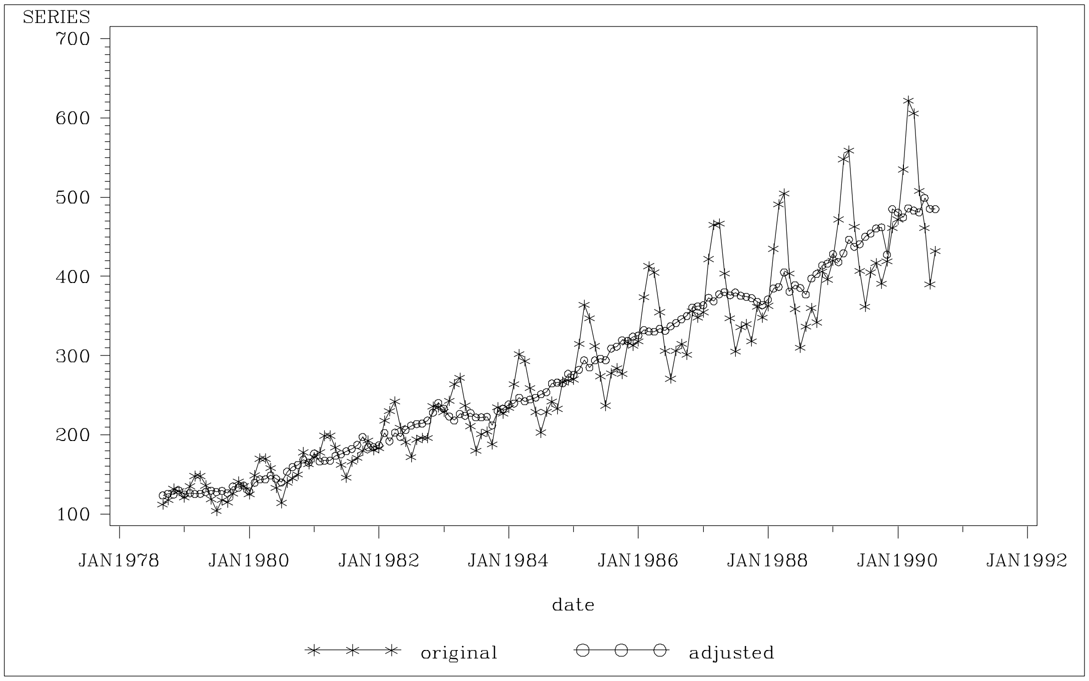
This example computes and plots the final estimates of the individual components for a monthly series. In the first plot, Output 31.1.1 an overlaid plot of the original and seasonally adjusted data is produced. The trend in the data is more evident in the seasonally adjusted data than in the original data. This trend is even more clear in Output 31.1.3, the plot of Table D12, the trend cycle. Note that both the seasonal factors and the irregular factors vary around 100, while the trend cycle and the seasonally adjusted data are in the scale of the original data.

From Output 31.1.2 the seasonal component appears to be slowly increasing, while no apparent pattern exists for the irregular series in Output 31.1.4.

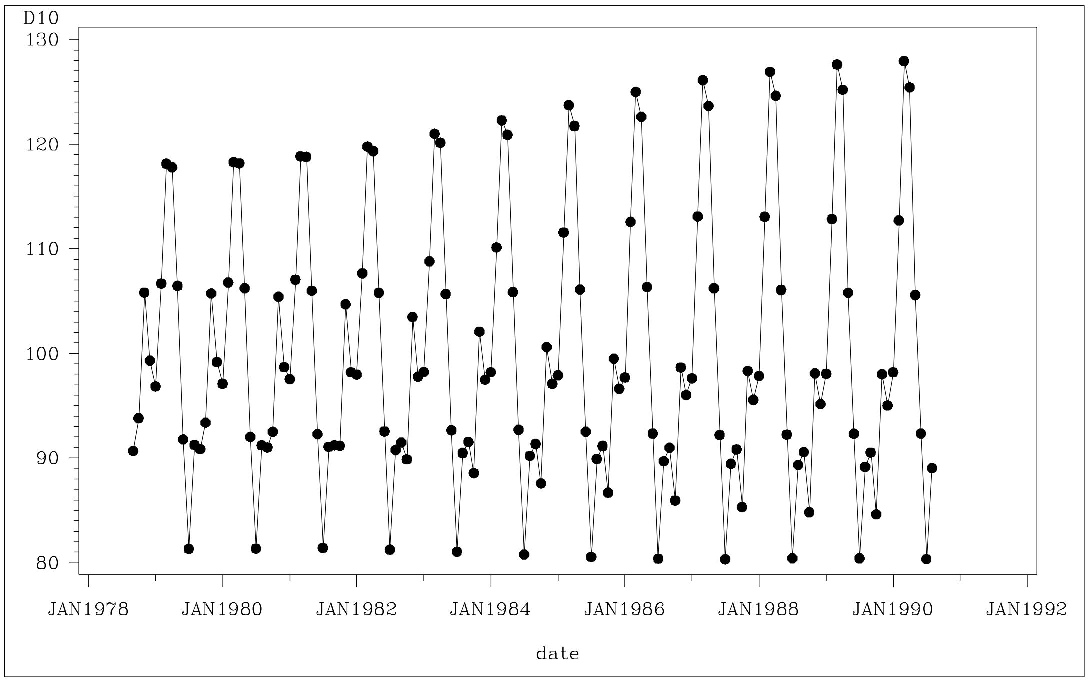
```
data sales;
   input sales @@;
   date = intnx( 'month', '01sep1978'd, _n_-1 );
   format date monyy7.;
   datalines;
112 118 132 129 121 135 148 148 136 119 104 118
115 126 141 135 125 149 170 170 158 133 114 140
145 150 178 163 172 178 199 199 184 162 146 166
171 180 193 181 183 218 230 242 209 191 172 194
196 196 236 235 229 243 264 272 237 211 180 201
204 188 235 227 234 264 302 293 259 229 203 229
242 233 267 269 270 315 364 347 312 274 237 278
284 277 317 313 318 374 413 405 355 306 271 306
315 301 356 348 355 422 465 467 404 347 305 336
340 318 362 348 363 435 491 505 404 359 310 337
360 342 406 396 420 472 548 559 463 407 362 405
417 391 419 461 472 535 622 606 508 461 390 432
run;

proc x11 data=sales noprint;
   monthly date=date;
   var sales;
   tables b1 d11;
   output out=out b1=series d10=d10 d11=d11
                  d12=d12 d13=d13;
run;

symbol1 i=join v='star';
symbol2 i=join v='circle';
legend1 label=none value=('original' 'adjusted');

proc gplot data=out;
   plot series * date = 1 d11 * date = 2
                        / overlay legend=lengend1;
run;

symbol1 i=join v=dot;
```

```
proc gplot data=out;
   plot ( d10 d12 d13 ) * date;
run;
```

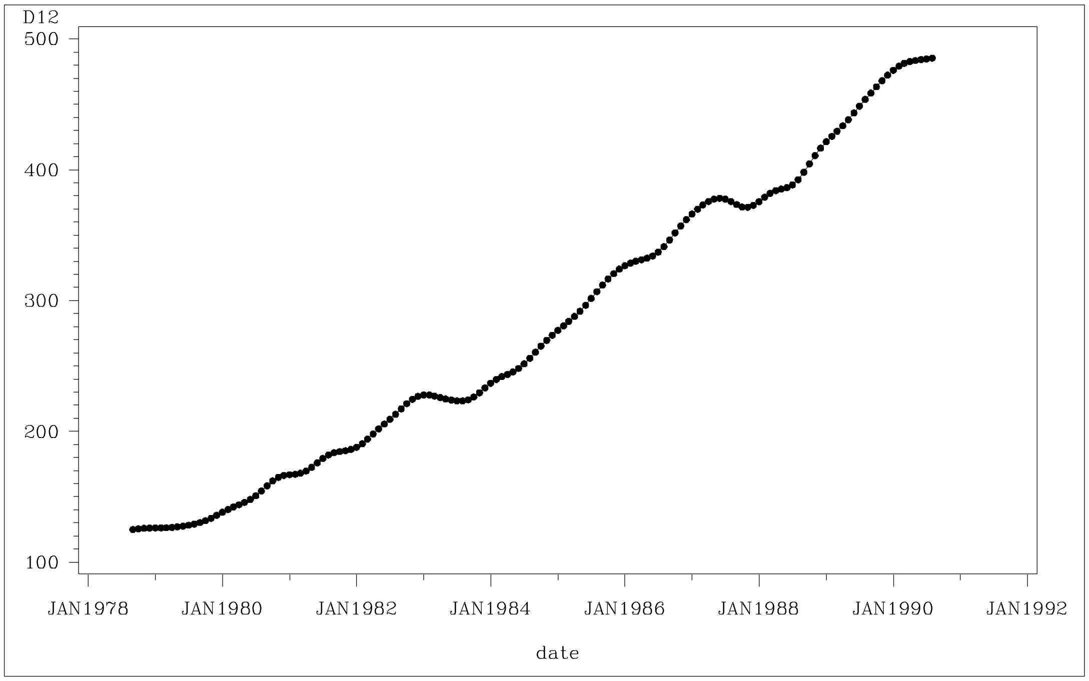
Output 31.1.1. Plot of Original and Seasonally Adjusted Data



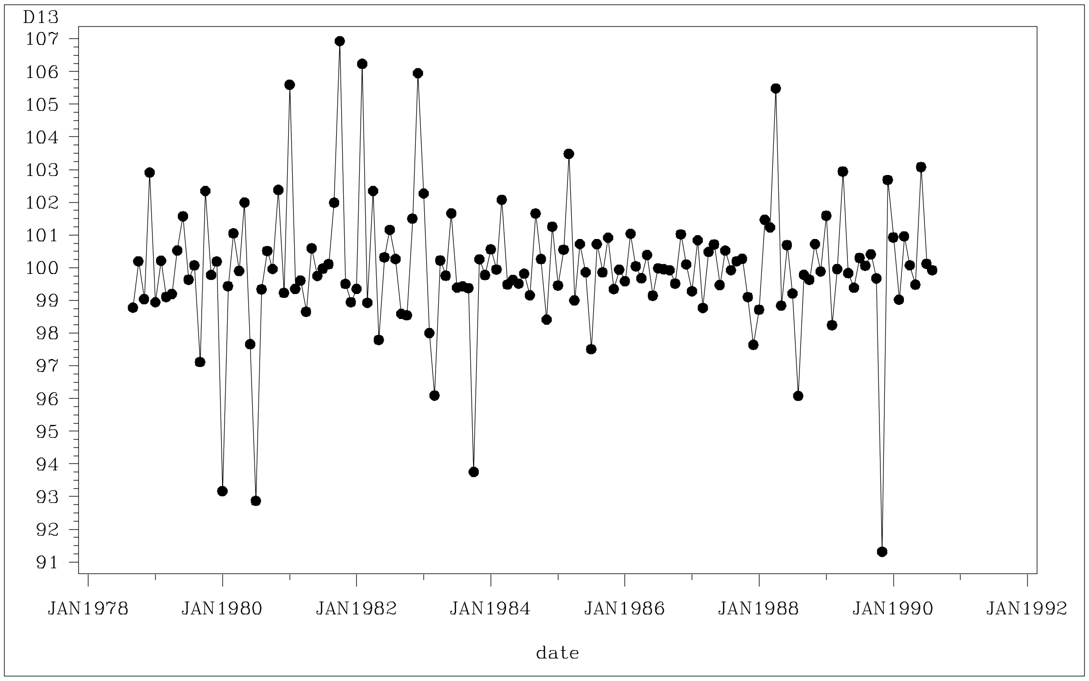
Output 31.1.2. Plot of D10, the Final Seasonal Factors



Output 31.1.3. Plot of D12, the Final Trend Cycle



Output 31.1.4. Plot of D13, the Final Irregular Series



Example 31.2. Components Estimation - Quarterly Data

This example is similar to Example 31.1, except quarterly data is used. Tables B1, the original series, and D11, the final seasonally adjusted series, are printed by the TABLES statement. The OUTPUT statement writes the listed tables to an output data set.

```
data quarter;
   input date yyq6. +1 fy35rr 5.2;
   format date yyq6.;
datalines;
1971Q1 6.59
1971Q2 6.01
1971Q3 6.51
1971Q4 6.18
1972Q1 5.52
1972Q2 5.59
1972Q3 5.84
1972Q4 6.33
1973Q1 6.52
1973Q2 7.35
1973Q3 9.24
1973Q4 10.08
1974Q1 9.91
1974Q2 11.15
1974Q3 12.40
1974Q4 11.64
1975Q1 9.94
1975Q2 8.16
1975Q3 8.22
1975Q4 8.29
1976Q1 7.54
1976Q2 7.44
1976Q3 7.80
1976Q4 7.28
run;

proc x11 data=quarter;
   var fy35rr;
   quarterly date=date;
   tables b1 d11;
   output out=out b1=b1 d10=d10 d11=d11 d12=d12 d13=d13;
run;
```

Output 31.2.1. Printed Output of PROC X11 Quarterly Example

```
                               The X11 Procedure

                        X-11 Seasonal Adjustment Program
                           U. S. Bureau of the Census
                     Economic Research and Analysis Division
                                November 1, 1968

          The X-11 program is divided into seven major parts.
           Part         Description
            A.  Prior adjustments, if any
            B.  Preliminary estimates of irregular component weights
                    and regression trading day factors
            C.  Final estimates of above
            D.  Final estimates of seasonal, trend-cycle and
                    irregular components
            E.  Analytical tables
            F.  Summary measures
            G.  Charts

                                Series - fy35rr
            Period covered - 1st Quarter 1971 to 4th Quarter 1976



                               The X11 Procedure

                        Seasonal Adjustment of - fy35rr

                              B1 Original Series
          Year         1st         2nd         3rd         4th       Total

          1971       6.590       6.010       6.510       6.180      25.290
          1972       5.520       5.590       5.840       6.330      23.280
          1973       6.520       7.350       9.240      10.080      33.190
          1974       9.910      11.150      12.400      11.640      45.100
          1975       9.940       8.160       8.220       8.290      34.610
          1976       7.540       7.440       7.800       7.280      30.060
          -----------------------------------------------------------
          Avg        7.670       7.617       8.335       8.300

                  Total:  191.53  Mean:  7.9804  S.D.:  1.9424
```

```
                               The X11 Procedure

                        Seasonal Adjustment of - fy35rr

                     D11 Final Seasonally Adjusted Series
          Year         1st         2nd         3rd         4th       Total

          1971       6.877       6.272       6.222       5.956      25.326
          1972       5.762       5.836       5.583       6.089      23.271
          1973       6.820       7.669       8.840       9.681      33.009
          1974      10.370      11.655      11.855      11.160      45.040
          1975      10.418       8.534       7.853       7.947      34.752
          1976       7.901       7.793       7.444       6.979      30.116
          -----------------------------------------------------------
          Avg        8.025       7.960       7.966       7.969

                  Total:  191.51  Mean:  7.9797  S.D.:  1.9059
```

Example 31.3. Outlier Detection and Removal

PROC X11 can be used to detect and replace outliers in the irregular component of a monthly or quarterly series.

The weighting scheme used in measuring the "extremeness" of the irregulars is developed iteratively; thus the statistical properties of the outlier adjustment method are unknown.

In this example, the data is simulated by generating a trend plus a random error. Two periods in the series were made "extreme" by multiplying one generated value by 2.0 and another by 0.10. The additive model is appropriate based on the way the data was generated. Note that the trend in the generated data was modeled automatically by the trend cycle component estimation.

The detection of outliers is accomplished by considering table D9, the final replacement values for extreme S-I ratios. This table indicates which observations had irregular component values more than FULLWEIGHT= standard deviation units from 0.0 (1.0 for the multiplicative model). The default value of the FULLWEIGHT= option is 1.5; a larger value would result in fewer observations being declared extreme.

In this example, FULLWEIGHT=3.0 is used to isolate the extreme inflated and deflated values generated in the data step. The value of ZEROWEIGHT= must be greater than FULLWEIGHT; it is given a value of 3.5.

A plot of the original and modified series, Output 31.3.2, shows that the deviation from the trend line for the modified series is greatly reduced compared with the original series.

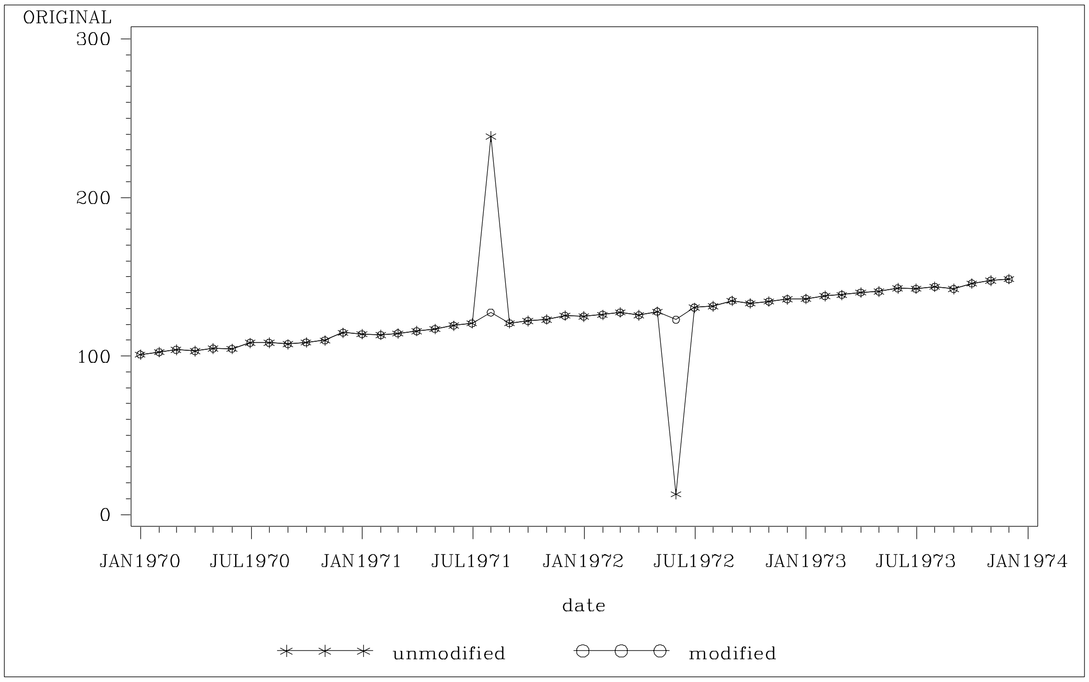
```
data a;
   retain seed 99831;
   do kk = 1 to 48;
      x = kk + 100 + rannor( seed );
      date = intnx( 'month', '01jan1970'd, kk-1 );
      if kk = 20 then x = 2 * x;
      else if kk = 30 then x = x / 10;
      output;
      end;
run;

proc x11 data=a;
   monthly date=date additive
           fullweight=3.0 zeroweight=3.5;
   var x;
   table d9;
   output out=b b1=original e1=e1;
run;

symbol1 i=join v=star;
symbol2 i=join v=circle;
legend1 label=none value=('unmodified' 'modified');

proc gplot data= b;
```

```
      plot original * date = 1 e1 * date = 2 / overlay legend=legend1;
      format date monyy7.;
   run;
```

Output 31.3.1. Detection of Extreme Irregulars

```
                           The X11 Procedure

                        Seasonal Adjustment of - x

            D9 Final Replacement Values for Extreme SI Ratios
   Year          JAN         FEB         MAR         APR         MAY         JUN

   1970         .           .           .           .           .           .
   1971         .           .           .           .           .           .
   1972         .           .           .           .           .       -10.671
   1973         .           .           .           .           .           .

            D9 Final Replacement Values for Extreme SI Ratios
   Year          JUL         AUG         SEP         OCT         NOV         DEC

   1970         .           .           .           .           .           .
   1971         .        11.180         .           .           .           .
   1972         .           .           .           .           .           .
   1973         .           .           .           .           .           .
```

Output 31.3.2. Plot of Modified and Unmodified Values



References

Bell, W.R., and Hillmer, S.C., (1984), "Issues Involved With the Seasonal Adjustment of Economic Time Series," *Journal of Business and Economic Statistics*, Vol. 2, No. 4.

Bobbit, L.G., and Otto, M.C., (1990), "Effects of Forecasts on the Revisions of Seasonally Adjusted Data Using the X-11 Adjustment Procedure," *Proceedings of the Business and Economic Statistics Section of the American Statistical Association*, pp. 449-453.

Buszuwski, J.A., (1987), "Alternative ARIMA Forecasting Horizons When Seasonally Adjusting Producer Price Data with X-11-ARIMA," *Proceedings of the Business and Economic Statistics Section of the American Statistical Association*, pp. 488-493.

Cleveland, W.S. and Devlin, S.J., (1980), "Calendar Effects in Monthly Time Series: Detection by Spectrum Analysis and Graphical Methods," *Journal of the American Statistical Association*, Vol. 75, No. 371, pp. 487-496.

Cleveland, W.P. and Tiao, G.C. (1976), "Decomposition of Seasonal Time Series: A Model for Census X-11 Program," *Journal of the American Statistical Association*, Vol. 71, No. 355.

Dagum, E.B. (1980), *The X-11-ARIMA Seasonal Adjustment Method*, Statistics Canada.

Dagum, E.B. (1982a), "Revisions of Time Varying Seasonal Filters," *Journal of Forecasting*, Vol. 1, pp. 173-187.

Dagum, E.B. (1982b), "The Effects of Asymmetric Filters on Seasonal Factor Revision," *Journal of the American Statistical Association*, Vol. 77, No. 380, pp. 732-738.

Dagum, E.B. (1982c), "Revisions of Seasonally Adjusted Data Due to Filter Changes," *Proceedings of the Business and Economic Section, the American Statistical Association*, pp. 39-45.

Dagum, E.S. (1983), "The X-11-ARIMA Seasonal Adjustment Method," Ottawa: Statistics Canada

Dagum, E.S. (1985), "Moving Averages," in *Encyclopedia of Statistical Sciences*, Vol. 5, eds. S. Kotz and N. L. Johnson, New York: John Wiley & Sons, Inc.

Dagum, E.S. and Laniel, N. (1987), "Revisions of Trend Cycle Estimators of Moving Average Seasonal Adjustment Method," *Journal of Business and Economic Statistics*, Vol. 5, No. 2, pp. 177-189.

Dagum, E.B. (1988), *The X11ARIMA/88 Seasonal Adjustment Method*, Statistics Canada.

Davies, N., Triggs, C.M., and Newbold, P. (1977), "Significance Levels of the Box-Pierce Portmanteau Statistic in Finite Samples," *Biometrika*, 64, 517-522.

Findley, D.F., Monsell, B.C., Shulman, H.B., and Pugh, M.G., (1990), "Sliding Spans Diagnostics for Seasonal and Related Adjustments," *Journal of the American Statistical Association*, Vol 85, No. 410, pps 345-355

Findley, D.F., Monsell, B.C., (1986) "New Techniques for Determining if a Time Series can be Seasonally Adjusted Reliably, and Their Application to U. S. Foreign Trade Series," in *Regional Econometric Modeling*, eds. M.R. Perryman and J. R. Schmidt, Amsterdam: Kluwer-Nijhoff, pp. 195-228.

Ghysels, E. (1990), "Unit Root Tests and the Statistical Pitfalls of Seasonal Adjustment: The Case of U.S. Post War Real GNP," *Journal of Business and Economic Statistics*, Vol. 8, No. 2, pp. 145-152.

Hannan, E.J., (1963), "The Estimation of Seasonal Variation in Economic Time Series," *Journal of the American Statistical Association*, Vol. 58, No. 301.

Huot, G., Chui, L., Higginson, J., and Gait, N., (1986), "Analysis of Revisions in the Seasonal Adjustment of Data Using X11ARIMA Model-Based Filters," International Journal of Forecasting, Vol. 2, pp. 217-229.

Laniel, N. (1985) "Design Criteria for the 13-term Henderson End-Weights," Working Paper, Methodology Branch, Ottawa: Statistics Canada

Ljung, G.M. and Box, G.E.P. (1978), "On a Measure of Lack of Fit in Time Series Models," *Biometrika*, 65, 297-303.

Lothian, J. and Morry M., (1978a), "Selection of Models for the Automated X-11-ARIMA Seasonal Adjustment Program," Statistics Canada.

Ladiray, D. and Quenneville B. (2001), “Seasonal Adjustment with the X-11 Method,” New York: Springer-Verlag.

Marris, S.N., (1960), "The Treatment of Moving Seasonality in Census Method II," *Seasonal Adjustments on Electronic Computers*, Organization for Economic Cooperation and Development.

Monsell, B.C., (1984), "The Substantive Changes in the X-11 Procedure of X-11-ARIMA," Bureau of the Census, Statistical Research Division, SRD Research Report Number Census/SRD/RR-84/10.

Nettheim, N.F., (1965), "A Spectral Study of Overadjustment for Seasonality," *1964 Proceedings of the Business and Economics Section; American Statistical Association*; republished as the Bureau of the Census Working Paper No. 21, 1965.

Pierce, D.A., (1980), "Data Revisions with Moving Average Seasonal Adjustment Procedures," *Journal of Econometrics*, 14, 95-114.

Shiskin, J., and Eisenpress, H., (1957), "Seasonal Adjustment by Electronic Computer Methods," JASA, Vol. 52, No. 280.

Shiskin, J., (1958), "Decomposition of Economic Time Series," Science, Vol. 128, No. 3338.

U.S. Bureau of the Census, (1965). "Estimating Trading Day Variation in Monthly Economic Time Series," Technical Paper No. 12. U.S. Government Printing Office, Washington, D.C.

U.S. Bureau of the Census, (1967). "The X-11 Variant of the Census Method II Seasonal Adjustment Program," Technical Paper No. 15, 1967 revision. U.S. Government Printing Office, Washington, D.C.

U.S. Bureau of the Census (1969), *X-11 Information for the User*, U.S. Department of Commerce, Washington, DC: Government Printing Office.

Chapter 32
The X12 Procedure

Chapter Contents

Chapter 32 The X12 Procedure

Overview

The X12 procedure, an adaptation of the U.S. Bureau of the Census X-12-ARIMA Seasonal Adjustment program (U.S. Bureau of the Census 2001c), seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set containing the adjusted time series and intermediate calculations.

The X-12-ARIMA program combines the capabilities of the X-11 program (Shiskin, Young, and Musgrave 1967) and the X-11-ARIMA/88 program (Dagum 1988) and also introduces some new features (Findley et al. 1998). One of the main enhancements involves the use of a regARIMA model, a regression model with ARIMA (autoregressive integrated moving average) errors. Thus, the X-12-ARIMA program contains methods developed by both the U.S. Census Bureau and Statistics Canada. In addition, the X-12-ARIMA automatic modeling routine is based on the TRAMO (Time series Regression with ARIMA noise, Missing values, and Outliers) method (Gomez and Maravall 1997a;b). The four major components of the X-12-ARIMA program are regARIMA modeling, model diagnostics, seasonal adjustment using enhanced X-11 methodology, and post-adjustment diagnostics. Statistics Canada's X-11 method fits an ARIMA model to the original series, then uses the model forecast to extend the original series. This extended series is then seasonally adjusted by the standard X-11 seasonal adjustment method. The extension of the series improves the estimation of the seasonal factors and reduces revisions to the seasonally adjusted series as new data become available.

Seasonal adjustment of a series is based on the assumption that seasonal fluctuations can be measured in the original series, O_t, t = 1, $\ldots$,n, and separated from trend-cycle, trading-day, and irregular fluctuations. The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated constantly or in an evolving fashion from year to year. The trend-cycle component, C_t, includes variation due to the long-term trend, the business cycle, and other long-term cyclical factors. The trading-day component, D_t, is the variation that can be attributed to the composition of the calendar. The irregular component, I_t, is the residual variation. Many economic time series are related in a multiplicative fashion ($O_t = S_t C_t D_t I_t$). Other economic series are related in an additive fashion ($O_t = S_t + C_t + D_t + I_t$). A seasonally adjusted time series, $C_t I_t$ or $C_t + I_t$, consists of only the trend-cycle and irregular components. For more details on seasonal adjustment with the X-11 method, refer to Ladiray and Quenneville (2001).

Experimental graphics are now available with the X12 procedure. For more information, see the "ODS Graphics" section on page 1951.

Getting Started

The most common use of the X12 procedure is to produce a seasonally adjusted series. Eliminating the seasonal component from an economic series facilitates comparison among consecutive months or quarters. A plot of the seasonally adjusted series is often more informative about trends or location in a business cycle than a plot of the unadjusted series.

The following example shows how to use PROC X12 to produce a seasonally adjusted series, C_tI_t, from an original series $O_t = S_tC_tD_tI_t$.

In the multiplicative model, the trend cycle component C_t keeps the same scale as the original series O_t, while S_t, D_t, and I_t vary around 1.0. In all displayed tables, these latter components are expressed as percentages and thus vary around 100.0 (in the additive case, they vary around 0.0). However, in the output data set, the data displayed as percentages will be expressed as the decimal equivalent and thus will vary around 1.0 in the multiplicative case.

The naming convention used in PROC X12 for the tables follows the convention used in the Census Bureau's X-12-ARIMA program; refer to *X-12-ARIMA Reference Manual* (U.S. Bureau of the Census 2001b) and *X-12-ARIMA Quick Reference for Unix* (U.S. Bureau of the Census 2001a). Also see the "Displayed Output/ODS Table Names/OUTPUT Tablename Keywords" section on page 1949 later in this chapter. The table names are outlined in Table 32.7 on page 1950.

The tables corresponding to parts A through C are intermediate calculations. The final estimates of the individual components are found in the D tables: table D10 contains the final seasonal factors, table D12 contains the final trend cycle, and table D13 contains the final irregular series. If you are primarily interested in seasonally adjusting a series without consideration of intermediate calculations or diagnostics, you need only to look at table D11, the final seasonally adjusted series. Tables in part E contain information regarding extreme values and changes in the original and seasonally adjusted series. The tables in part F are seasonal adjustment quality measures. Spectral analysis is performed in part G. For further information concerning the tables produced by the X11 statement, refer to Ladiray and Quenneville (2001).

Basic Seasonal Adjustment

Suppose that you have monthly retail sales data starting in September 1978 in a SAS data set named SALES. At this point, you do not suspect that any calendar effects are present, and there are no prior adjustments that need to be made to the data.

In this simplest case, you need only specify the DATE= variable in the PROC X12 statement and request seasonal adjustment in the X11 statement. The results of the seasonal adjustment are in table D11 (the final seasonally adjusted series) in the displayed output as shown in Figure 32.1.

```
data sales;
   set sashelp.air;
   sales = air;
```

```
      date = intnx( 'month', '01sep78'd, _n_-1 );
      format date monyy.;
   run ;


   proc x12 data=sales date=date;
      var sales;
      x11;
   run ;
```

```
                                 The X12 Procedure

              Table D 11:  Final seasonally adjusted data
                           For variable sales
   Year         JAN        FEB        MAR        APR        MAY        JUN
                JUL        AUG        SEP        OCT        NOV        DEC      Total

   1978          .          .          .          .          .          .
                 .          .      124.560    124.649    124.920    129.002    503.131
   1979      125.087    126.759    125.252    126.415    127.012    130.041
             128.056    129.165    127.182    133.847    133.199    135.847    1547.86
   1980      128.767    139.839    143.883    144.576    148.048    145.170
             140.021    153.322    159.128    161.614    167.996    165.388    1797.75
   1981      175.984    166.805    168.380    167.913    173.429    175.711
             179.012    182.017    186.737    197.367    183.443    184.907    2141.71
   1982      186.080    203.099    193.386    201.988    198.322    205.983
             210.898    213.516    213.897    218.902    227.172    240.453    2513.69
   1983      231.839    224.165    219.411    225.907    225.015    226.535
             221.680    222.177    222.959    212.531    230.552    232.565    2695.33
   1984      237.477    239.870    246.835    242.642    244.982    246.732
             251.023    254.210    264.670    266.120    266.217    276.251    3037.03
   1985      275.485    281.826    294.144    286.114    293.192    296.601
             293.861    309.102    311.275    319.239    319.936    323.663    3604.44
   1986      326.693    330.341    330.383    330.792    333.037    332.134
             336.444    341.017    346.256    350.609    361.283    362.519    4081.51
   1987      364.951    371.274    369.238    377.242    379.413    376.451
             378.930    375.392    374.940    373.612    368.753    364.885    4475.08
   1988      371.618    383.842    385.849    404.810    381.270    388.689
             385.661    377.706    397.438    404.247    414.084    416.486    4711.70
   1989      426.716    419.491    427.869    446.161    438.317    440.639
             450.193    454.638    460.644    463.209    427.728    485.386    5340.99
   1990      477.259    477.753    483.841    483.056    481.902    499.200
             484.893    485.245        .          .          .          .      3873.15
   ---------------------------------------------------------------------------
   Avg       277.330    280.422    282.373    286.468    285.328    288.657
             288.389    291.459    265.807    268.829    268.774    276.446

                     Total:  40323  Mean:  280.02  S.D.:  111.31
                                     Min:   124.56  Max:   499.2
```

Figure 32.1. Basic Seasonal Adjustment

You can compare the original series (table A1) and the final seasonally adjusted series (table D11) by plotting them together as shown in Figure 32.2. These tables are requested in the OUTPUT statement and are written to the OUT= data set. Note that the default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name.

```
proc x12 data=sales date=date;
   var sales;
   x11;
   output out=out a1 d11;
run ;

axis1 label=none;
axis2 label=none;
symbol1 i=join v='star' c=red;
symbol2 i=join v='circle' c=blue;
legend1 label=none value=('original' 'adjusted');

proc gplot data=out;
   plot sales_A1    * date = 1
        sales_D11   * date = 2
      / overlay legend=legend1 haxis=axis1 vaxis=axis2;
run;
```

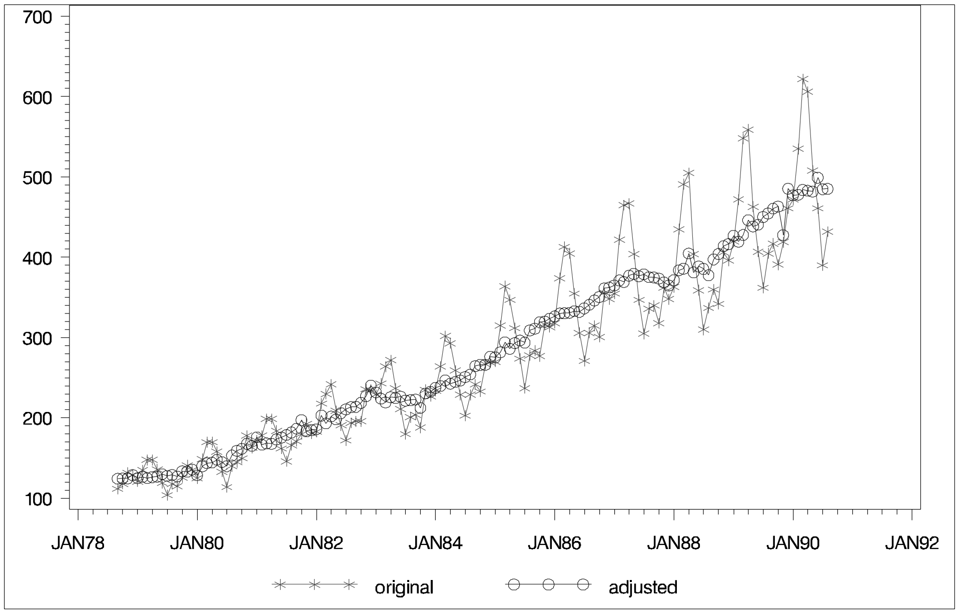


Figure 32.2. Plot of Original and Seasonally Adjusted Data

Syntax

The X12 procedure uses the following statements:

PROC X12 *options*;
  VAR *variables*;
  BY *variables*;
  ID *variables*;
  TRANSFORM *options*;
  IDENTIFY *options*;
  AUTOMDL *options*;
  OUTLIER *options*;
  REGRESSION *options*;
  ARIMA *options*;
  ESTIMATE *options*;
  X11 *options*;
  FORECAST *options*;
  OUTPUT *options*;

The PROC X12 statements perform basically the same function as the Census Bureau's X-12-ARIMA specs. *Specs* or specifications are used in X-12-ARIMA to control the computations and output. The PROC X12 statement performs some of the same functions as the Series spec in the Census Bureau's X-12-ARIMA software. The TRANSFORM, IDENTIFY, AUTOMDL, OUTLIER, REGRESSION, ARIMA, ESTIMATE, X11, and FORECAST statements are designed to perform the same functions as the corresponding X-12-ARIMA *specs*, although full compatibility is not yet available. The Census Bureau documentation *X-12-ARIMA Reference Manual* (U.S. Bureau of the Census 2001b) can provide added insight to the functionality of these statements.

Functional Summary

The following table outlines the options available for the X12 procedure classified by function.

Description	Statement	Option
Data Set Options		
specify input data set	PROC X12	DATA=
write table values to an output data set	OUTPUT	OUT=
Printing Control Options		
suppress all printed output	PROC X12	NOPRINT

Description	Statement	Option
Date Information Options		
specify the date variable	PROC X12	DATE=
specify the date of the first observation	PROC X12	START=
specify the beginning date of the subset	PROC X12	SPAN=(mmmyy)
	PROC X12	SPAN=('yyQq')
specify the ending date of the subset	PROC X12	SPAN=(, mmmyy)
	PROC X12	SPAN=(, 'yyQq')
specify monthly time series	PROC X12	INTERVAL=MONTH
specify monthly time series	PROC X12	SEASONS=12
specify quarterly time series	PROC X12	INTERVAL=QTR
specify quarterly time series	PROC X12	SEASONS= 4
Declaring the Role of Variables		
specify BY-group processing	BY	
specify identifying variables	ID	
specify the variables to be seasonally adjusted	VAR	
Controlling the Table Computations		
transform or prior adjust the series	TRANSFORM	FUNCTION=
transform or prior adjust the series	TRANSFORM	POWER=
use differencing to identify the ARIMA part of the model	IDENTIFY	
use X-12-ARIMA TRAMO-based method to choose a model	AUTOMDL	
specify automatic outlier detection	OUTLIER	
specify regression information	REGRESSION	PREDEFINED=
specify the ARIMA part of the model	ARIMA	MODEL=
estimate the regARIMA model specified by the REGRESSION and ARIMA statements	ESTIMATE	
specify seasonal adjustment	X11	
specify the number of forecasts	FORECAST	LEAD=

PROC X12 Statement

PROC X12 *options;*

The PROC X12 statement provides information about the time series to be processed by PROC X12. Either the START= or the DATE= option must be specified.

The original series is displayed in table A1. If there are missing values in the original series, and a regARIMA model is specified or automatically selected, then table MV1 is displayed. Table MV1 contains the original series with missing values replaced by the predicted values from the fitted model. Table B1 will be displayed when the original data is altered, for example, through an ARIMA model estimation, prior adjustment factors, an irregular regression, or the series is extended with forecasts.

Although the X-12-ARIMA method will handle missing values, there are some restrictions. In order for Proc X12 to process the series, no month or quarter may contain missing values for all years. For instance, if the third quarter contained only missing values for all years, then processing will be skipped for that series. In addition, if more than half the values for a month or a quarter are missing, then a warning message will be displayed in the log file, and other errors may occur later in processing. If a series contains many missing values, the user should consider other methods of missing value replacement.

The following options can appear in the PROC X12 statement.

DATA= *SAS-data-set*
specifies the input SAS data set used. If this option is omitted, the most recently created SAS data set is used.

DATE= *variable*
DATEVAR= *variable*
specifies a variable that gives the date for each observation. Unless specified in the SPAN= option, the starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS datetime values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly in ascending order. If the INTERVAL= or SEASONS= option is specified, its values must agree with the values of the date variable. If neither the INTERVAL= or SEASONS= option is specified, then the procedure tries to determine the type of data from the values of the date variable. This variable is automatically added to the OUTPUT= data set if one is requested and is extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

START= *mmmyy*
START= *'yyQq'*
STARTDATE= *mmmyy*
STARTDATE= *'yyQq'*
gives the date of the first observation. Unless the SPAN= option is used, the starting and ending dates are the dates of the first and last observations, respectively. Either this option or the DATE= option is required. When using this option, use either the INTERVAL= or SEASONS= option to specify monthly or quarterly data. If neither the INTERVAL= or SEASONS= is present, monthly data are assumed. Note that for a quarterly date, the specification must be enclosed in quotes. A 4-digit year can be specified, but if a 2-digit year is given, the value specified in the YEARCUTOFF= SAS system option applies. When using this option with BY processing, the start date will be applied to the first observation in each BY group.

SPAN= (*mmmyy*,*mmmyy***)**
SPAN= (*'yyQq'*,*'yyQq'***)**
gives the dates of the first and last observations to define a subset for processing. A single date in parentheses is interpreted to be the starting date of the subset. To specify only the ending date, use SPAN=(,*mmmyy*). If the starting or ending date is omitted, then the first or last date, respectively, of the input data set is assumed. A 4-digit year can be specified, but if a 2-digit year is given, the value specified in the YEARCUTOFF= SAS system option applies.

INTERVAL= *interval*

specifies the frequency of the input time series. If the input data consist of quarterly observations, then INTERVAL=QTR should be used. If the input data consist of monthly observations, then INTERVAL=MONTH should be used. If the INTERVAL= option is not specified and SEASONS=4, then INTERVAL=QTR is assumed; likewise, SEASONS=12 implies INTERVAL=MONTH. If both are used, the values should not be conflicting. If neither the INTERVAL= nor SEASONS= option is specified and the START= option is specified, then the data are assumed to be monthly. If a date variable is specified using the DATE= option, it is not necessary to specify the INTERVAL= or SEASONS= option; however, if specified, the values of the INTERVAL= or SEASONS= option should not be in conflict with the values of the date variable.

SEASONS= *period*

specifies the number of observations in a seasonal cycle. If the SEASONS= option is not specified and INTERVAL=QTR, then SEASONS=4 is assumed. If the SEASONS= option is not specified and INTERVAL=MONTH, then SEASONS=12 is assumed. If the SEASONS= option is specified, its value should not conflict with the values of the INTERVAL= option or the values of the date variable. See the preceding descriptions for the START=, DATE=, and INTERVAL= options for more details.

NOTRIMMISS

By default leading and trailing missing values are trimmed from the series. NOTRIMMISS suppresses the default. If NOTRIMMISS is used, Proc X12 will automatically generate missing value regressors for any missing value within the span of the series, including leading and trailing missing values.

NOPRINT

suppresses any printed output.

BY Statement

BY *variables;*

A BY statement can be used with PROC X12 to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input DATA= data set to be sorted in order of the BY variables.

ID Statement

ID *variables;*

If you are creating an output data set, use the ID statement to put values of the ID variables, in addition to the table values, into the output data set. The ID statement has no effect when an output data set is not created. If the DATE= variable is specified in the Proc X12 statement, this variable is included automatically in the OUTPUT data set. If no DATE= variable is specified, the variable _DATE_ is added.

The date variable (or _DATE_) values outside the range of the actual data (from forecasting) are extrapolated, while all other ID variables are missing.

ARIMA Statement

ARIMA *options;*

The ARIMA statement specifies the ARIMA part of the regARIMA model. This statement defines a pure ARIMA model if the regression statement is omitted. The ARIMA part of the model can include multiplicative seasonal factors.

The following option can appear in the ARIMA statement.

MODEL= ((*p d q* **)(** *P D Q* **)** *s* **)**

specifies the ARIMA model. The format follows standard Box-Jenkins notation (Box, Jenkins, and Reinsel 1994). The nonseasonal AR and MA orders are given by *p* and *q*, respectively, while the seasonal AR and MA orders are given by *P* and *Q*. The number of differences and seasonal differences are given by *d* and *D*, respectively. The notation (*p d q*) and (*P D Q*) can also be specified as (*p*, *d*, *q*) and (*P*, *D*, *Q*). The maximum lag of any AR or MA parameter is 36. The maximum value of a difference order, *d* or *D*, is 144. All values for *p*, *d*, *q*, *P*, *D*, and *Q* should be nonnegative integers. The lag corresponding to seasonality is *s*. *s* should be a positive integer. If *s* is omitted, it is set equal to the value used in the PROC X12 SEASONS= statement.

For example,

```
proc x12 data=ICMETI seasons=12 start=jan1968;
   arima model=((2,1,1)(1,1,0));
```

specifies an ARIMA (2,1,1)(1,1,0)12 model.

ESTIMATE Statement

ESTIMATE *options;*

The ESTIMATE statement estimates the regARIMA model specified by the REGRESSION and ARIMA statements. Estimation output includes point estimates and standard errors for all estimated AR, MA, and regression parameters; the maximum likelihood estimate of the variance σ^2; *t* statistics for individual regression parameters; χ^2 statistics for assessing the joint significance of the parameters associated with certain regression effects (if included in the model); and likelihood-based model selection statistics (if the exact likelihood function is used). The regression effects for which χ^2 statistics are produced are fixed seasonal effects.

Tables displayed in association with estimation are Exact ARMA Likelihood Estimation Iteration Tolerances, Average absolute percentage error in within-sample forecasts:, ARMA Iteration History, AR/MA Roots, Exact ARMA Likelihood Estimation Iteration Summary, Regression Model Parameter Estimates, Chi-squared Tests for Groups of Regressors, Exact ARMA Maximum Likelihood Estimation, and Estimation Summary.

The following options can appear in the ESTIMATE statement.

MAXITER= *value*

Specifies the maximum number allowed of ARMA iterations (nonlinear iterations for estimating the AR and MA parameters). For models with regression variables, this limit applies to the total number of ARMA iterations over all IGLS iterations. For models without regression variables, this is the maximum number of iterations allowed for the single set of ARMA iterations The default is MAXITER=200.

TOL= *value*

Specifies the convergence tolerance for the nonlinear estimation. Absolute changes in the log-likelihood are compared to TOL to check convergence of the estimation iterations. For models with regression variables, TOL is used to check convergence of the IGLS iterations (where the regression parameters are reestimated for each new set of AR and MA parameters). For models without regression variables there are no IGLS iterations, and TOL is then used to check convergence of the nonlinear iterations used to estimate the AR and MA parameters. The default value is TOL=0.00001.

ITPRINT

Specifies that the Iterations History table will be displayed. This includes detailed output for estimation iterations, including log-likelihood values and parameters, and counts of function evaluations and iterations. It is useful to examine the Iterations History table when errors occur within estimation iterations. By default, only successful iterations are displayed, unless PRINTERR is specified. An unsuccessful iteration is an iteration which is restarted due to a problem such as a root inside the unit circle. Successful iterations will have a status of 0. If restarted iterations are displayed, a note at the end of the table will give definitions for status codes that indicate a restarted iteration. For restarted iterations, the number of function evaluations and the number of iterations will be -1, which will be displayed as missing. If regression parameters are included in the model, then both IGLS and ARMA iterations will be included in the table. The number of function evaluations is a cumulative total.

PRINTERR

Use of PRINTERR either causes restarted iterations to be included in the Iterations History table (if ITPRINT is specified), or creates the Restarted Iterations table (if ITPRINT is not specified). Whether or not PRINTERR is specified, a WARNING message will be printed to the log file if any iteration is restarted during estimation.

FORECAST Statement

FORECAST *options;*

The FORECAST statement is used to forecast the time series using the estimated model. The output contains point forecast and forecast statistics for the transformed and original series.

The following option can appear in the FORECAST statement.

LEAD= *value*

Specifies the number of periods ahead to forecast. The default is the number of periods in a year (4 or 12), and the maximum is 60.

Forecasts and Standard Errors Tables are displayed in association with the FORECAST statement. Confidence limits are also included. If the data is transformed, then two tables will be displayed, one table for the original data, and one table for the transformed data.

IDENTIFY Statement

IDENTIFY *options;*

The IDENTIFY statement must be used to produce plots of the sample Autocorrelation Functions (ACFs) and Partial Autocorrelation Functions (PACFs) for identifying the ARIMA part of a regARIMA model. Sample ACFs and PACFs are produced for all combinations of the nonseasonal and seasonal differences of the data specified by the DIFF and SDIFF options. If the REGRESSION statement is present, the ACFs and PACFs are calculated for the specified differences of a series of regression residuals. If the REGRESSION statement is not present, the ACFs and PACFs are calculated for the specified differences of the original data.

Tables printed in association with identification are “Autocorrelation of Model Residuals” and “Partial Autocorrelation of Model Residuals”. If the REGRESSION statement is present, then the “Regression Model Parameter Estimates” table will also be available.

The following options can appear in the IDENTIFY statement.

DIFF= *(order, order, order)*
specifies orders of nonseasonal differencing. The value 0 specifies no differencing, the value 1 specifies one nonseasonal difference $(1 - B)$, the value 2 specifies two nonseasonal differences $(1 - B)^2$, and so forth. The ACFs and PACFs are produced for all orders of nonseasonal differencing specified, in combination with all orders of seasonal differencing specified in the SDIFF= option. The default is DIFF=(0). You can specify up to three values for nonseasonal differences.

SDIFF= *(order, order, order)*
specifies orders of seasonal differencing. The value 0 specifies no seasonal differencing, the value 1 specifies one seasonal difference $(1 - B^s)$, the value 2 specifies two seasonal differences $(1 - B^s)^2$, and so forth. Here the value for *s* will correspond to the value of the SEASONS= option in the PROC X12 statement. The value of SEASONS= is supplied explicitly or is implicitly supplied through the INTERVAL= option or the values of the DATE= variable. The ACFs and PACFs are produced for all orders of seasonal differencing specified, in combination with all orders of nonseasonal differencing specified in the DIFF= option. The default is SDIFF=(0). You can specify up to three values for seasonal differences.

For example,

```
identify diff=(1) sdiff=(0, 1);
```

produces ACFs and PACFs for two models: $(1 - B)$ and $(1 - B)(1 - B^s)$.

PRINTREG
causes the "Regression Model Parameter Estimates" table to be printed if the REGRESSION statement is present. By default, the table is not printed.

AUTOMDL Statement

AUTOMDL *options;*

The AUTOMDL statement is used to invoke the automatic model selection procedure of the X-12-ARIMA method. This method is based largely on the TRAMO (Time series Regression with ARIMA noise, Missing values, and Outliers) method by Gomez and Maravall (1997a;b). If the AUTOMDL statement is used without the OUTLIER statement, then only missing values will be identified. If the AUTOMDL and the OUTLIER statements are used, then both missing values and outliers will be identified. If both the AUTOMDL statement and the ARIMA statement are present, the ARIMA statement will be ignored. The ARIMA statement specifies the model, while the AUTOMDL statement allows the X12 procedure to select the model.

When AUTOMDL is specified, the X12 procedure will compare a model selected using a TRAMO method to a default model. The TRAMO method is implemented first, and involves two parts: identifying the orders of differencing and identifying the ARIMA model. The table "ARIMA Estimates for Unit Root Identification" provides details regarding the identification of the orders of differencing, while "Results of Unit Root Test for Identifying Orders of Differencing" shows the orders of differencing selected by TRAMO. The table "Models estimated by Automatic ARIMA Model Selection procedure" provides details regarding the TRAMO automatic model selection, and the table "Best Five ARIMA Models Chosen by Automatic Modeling" ranks the best five models estimated using the TRAMO method. The next available table, "Comparison of Automatically Selected Model and Default Model," compares the model selected by the TRAMO method to a default model. At this point, if the Default Model is selected over the TRAMO model, Proc X12 will display a note. No note will be displayed if the TRAMO model is selected. Proc X12 will then perform checks for unit roots, overdifferencing, and insignificant ARMA coefficients. If the model is changed due to any of these tests, a note will be displayed. The last table, "Final Automatic Model Selection," shows the results of automatic model selection.

The following options can appear in the AUTOMDL statement.

MAXORDER= *(nonseasonal order, seasonal order)*
specifies the maximum orders of nonseasonal and seasonal ARMA polynomials for the automatic ARIMA model identification procedure. The maximum order for the nonseasonal ARMA parameters should be between 1 and 4; the maximum order for the seasonal ARMA should be 1 or 2.

DIFFORDER= *(nonseasonal order, seasonal order)*
specifies the fixed orders of differencing to be used in the automatic ARIMA model identification procedure. When DIFFORDER is used, only the AR and MA orders are automatically identified. Acceptable values for the regular differencing orders are 0, 1 and 2; acceptable values for the seasonal differencing orders are 0 and 1. If the MAXDIFF option is also specified, the DIFFORDER option will be ignored.

There are no default values for DIFFORDER. If neither the DIFFORDER nor the MAXDIFF option is specified, the default is MAXDIFF=(2,1).

MAXDIFF= *(nonseasonal order, seasonal order)*

specifies the maximum orders of regular and seasonal differencing for the automatic identification of differencing orders. When MAXDIFF is specified, the differencing orders will first be identified, and then the AR and MA orders will be identified. Acceptable values for the regular differencing orders are 1 and 2; the only acceptable value for the seasonal differencing orders is 1. If the DIFFORDER option is also specified, the DIFFORDER option will be ignored. If neither option is specified, the default is MAXDIFF=(2,1).

PRINT= UNITROOTTEST
PRINT= AUTOCHOICE
PRINT= UNITROOTTESTMDL
PRINT= AUTOCHOICEMDL
PRINT= BEST5MODEL

lists the tables to be displayed in the output. AUTOCHOICE is displayed by default. The default tables are titled “Comparison of Automatically Selected Model and Default Model” and “Final Automatic Model Selection.” “Comparison of Automatically Selected Model and Default Model” compares a default model to the model chosen by the TRAMO-based automatic modeling method. “Final Automatic Model Selection” indicates which model has been chosen automatically.

Unless the nonseasonal and seasonal differences are specified using the DIFFORDER option, AUTOMDL automatically identifies the orders of differencing. PRINT=UNITROOTTEST causes the table titled “Results of Unit Root Test for Identifying Orders of Differencing” to be printed; this table displays the orders which were automatically selected by AUTOMDL.

PRINT=UNITROOTMDL displays the table titled “ARIMA Estimates for Unit Root Identification.” This table summarizes the various models that were considered by the TRAMO automatic selection method while identifying the orders of differencing and the statistics associated with those models. The unit root identification method will first attempt to obtain the coefficients using the Hannan-Rissanen method. If Hannan-Rissanen estimation cannot be performed, the algorithm will attempt to obtain the coefficients using conditional likelihood estimation.

PRINT=AUTOCHOICEMDL displays the table “Models Estimated by Automatic ARIMA Model Selection Procedure.” This table summarizes the various models that were considered by the TRAMO automatic model selection method and their measures of fit.

PRINT=BEST5MODEL displays the table “Best Five ARIMA Models Chosen by Automatic Modeling.” This table ranks the five best models that were considered by the TRAMO automatic modeling method.

BALANCED
specifies that the automatic model procedure will have a preference for balanced models (sum of AR, differencing, and seasonal differencing orders = sum of MA and seasonal MA orders). Specifying BALANCED gives the same preference as the TRAMO program. If BALANCED is not specified, all models will be given equal consideration.

HRINITIAL
specifies that Hannan-Rissanen estimation is done before exact maximum likelihood estimation to provide initial values. If HRINITIAL is specified, then models for which the Hannan-Rissanen estimation has an unacceptable coefficient will be rejected.

ACCEPTDEFAULT
specifies that the default model is chosen if its Ljung-Box Q is acceptable.

LJUNGBOXLIMIT= *value*
specifies acceptance criteria for confidence coefficient of the Ljung-Box Q statistic. If the Ljung-Box Q for a final model is greater than this value, the model is rejected, the outlier critical value is reduced, and outlier identification is redone with the reduced value (see reducecv option). The value specified must be greater than 0.0 and less than 1.0. The default value is 0.95.

REDUCECV= *value*
specifies the percentage that the outlier critical value will be reduced when a final model is found to have an unacceptable confidence coefficient for the Ljung-Box Q statistic. This value should be between 0 and 1. The default value is 0.14286.

ARMACV= *value*
specifies the threshold value for *t*-statistics of ARMA coefficients for tests of model parsimony. Insignificant ARMA coefficients whose *t*-values have an absolute value less than this value will be set to zero. An ARMA coefficient is considered to be insignificant if it is below 0.15 for 150 or less observations and below 0.1 for more than 150 observations. This value is also used as the critical value for testing for adding a constant to the regression model. A constant regressor will be added if the *t*-statistic is below the critical value. Note that if a constant regressor is added to the model, and then the ARIMA model changes, then the *t*-statistic will also change. This value should be greater than zero. The default value is 1.0.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set tablename1 tablename2 ... ;*

The OUTPUT statement creates an output data set containing specified tables. The data set is named by the OUT= option.

OUT= *SAS-data-set*
If the OUT= option is omitted, the SAS System names the new data set using the default DATA*n* convention.

For each table to be included in the output data set, you must specify the X12 *tablename* keyword. The keyword corresponds to the title label used by the Census Bureau

X12-ARIMA software. Currently available tables are A1, A2, A6, A8, A8AO, A8LS, A8TC, B1, C17, C20, D1, D7, D8, D9, D10, D10D, D11, D12, D13, D16, D16B, D18, E5, E6, E7, and MV1. If no table is specified, table A1 will be output to the dataset by default.

The tablename keywords that can be used in the OUTPUT statement are listed in the "Displayed Output/ODS Table Names/OUTPUT Tablename Keywords" section on page 1949. The following is an example of a VAR statement and an OUTPUT statement:

```
var sales costs;
output out=out_x12  b1 d11;
```

Note that the default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name. The variable sales_B1 contains the table B1 values for the variable sales, the variable costs_B1 contains the table B1 values for costs, while the table D11 values for sales are contained in the variable sales_D11, and the variable costs_D11 contains the table D11 values for costs. If necessary, the variable name is shortened so that the table name can be added. Currently, you cannot specify the output variable names. If DATE= is specified in the PROC X12 statement, then that variable is included in the output data set. Otherwise, a variable named _DATE_ is the date identifier.

OUTLIER Statement

OUTLIER *options;*

The OUTLIER statement specifies that the X12 procedure perform automatic detection of additive (point) outliers, temporary change outliers, level shifts, or any combination of the three using the specified model. After outliers are identified, the appropriate regression variables are incorporated into the model as "Automatically Identified Outliers," and the model is reestimated. This procedure is repeated until no additional outliers are found.

The OUTLIER statement also identifies potential outliers and lists them in the table "Potential Outliers" in the displayed output. Potential outliers are identified by decreasing the critical value by 0.5.

In the output, the default initial critical values used for outlier detection in a given analysis are displayed in the table "Critical Values to Use in Outlier Detection." Outliers that are detected and incorporated into the model are displayed in the output in the table "Regression Model Parameter Estimates," where the regression variable is listed as "Automatically Identified."

The following options can appear in the OUTLIER statement.

TYPE= NONE
TYPE= *(outlier types)*

lists the outlier types to be detected by the automatic outlier identification method. TYPE=NONE turns off outlier detection. The valid outlier types are AO, LS, and TC. The default is TYPE=(AO LS).

CV= *value*

specifies an initial critical value to use for detection of all types of outliers. The absolute value of the *t*-statistic associated with an outlier parameter estimate is compared with the critical value to determine the significance of the outlier. If the CV= option is not specified, then the default initial critical value is computed using a formula presented by Ljung (1993), which is based on the number of observations or model span used in the analysis. Table 32.1 gives default critical values for various series lengths. Raising the critical value decreases the sensitivity of the outlier detection routine, and may reduce the number of observations treated as outliers. The automatic model identification process may lower the critical value by a certain percentage, if the automatic model identification process fails to identify an acceptable model.

Table 32.1. Default Critical Values for Outlier Identification

Number of Observations	Outlier Critical Value
1	1.96
2	2.24
3	2.44
4	2.62
5	2.74
6	2.84
7	2.92
8	2.99
9	3.04
10	3.09
11	3.13
12	3.16
24	3.42
36	3.55
48	3.63
72	3.73
96	3.80
120	3.85
144	3.89
168	3.92
192	3.95
216	3.97
240	3.99
264	4.01
288	4.03
312	4.04
336	4.05
360	4.07

AOCV= *value*

specifies a critical value to use for additive (point) outliers. If AOCV is specified, this value will override any default critical value for AO outliers. See the CV= option for more details.

LSCV= *value*

specifies a critical value to use for level shift outliers. If LSCV is specified, this value will override any default critical value for LS outliers. See the CV= option for more details.

TCCV= *value*

specifies a critical value to use for temporary change outliers. If TCCV is specified, this value will override any default critical value for TC outliers. See the CV= option for more details.

REGRESSION Statement

REGRESSION *options;*

The REGRESSION statement includes regression variables in a regARIMA model or specifies regression variables whose effects are to be removed by the IDENTIFY statement to aid in ARIMA model identification. Predefined regression variables are selected with the PREDEFINED option. The currently available predefined variables are listed below. Table A6 provides information related to trading-day effects. Tables A8, A8AO, A8LS, and A8TC provide information related to outlier factors. You should note that missing values in the span of an input series automatically create missing value regressors. See the NOTRIMMISS option of the Proc X12 statement and the "Missing Values" section on page 1949 later in this chapter for further details regarding missing values. Combining your model with additional predefined regression variables may result in a singularity problem. If a singularity occurs, then you may need to alter either the model or the choices of the predefined regressors in order to successfully perform the regression.

In order to seasonally adjust a series using a regARIMA model, the factors derived from the regression coefficients must be the same type as factors generated by the seasonal adjustment procedure, so that combined adjustment factors can be derived and adjustment diagnostics can be generated. If the regARIMA model is applied to a log-transformed series, the regression factors are expressed in the form of ratios, which match seasonal factors generated by the multiplicative (or log-additive) adjustment modes. Conversely, if the regARIMA model is fit to the original series, the regression factors are measured on the same scale as the original series, which match seasonal factors generated by the additive adjustment mode. You should note that the default transformation (no transformation) and the default seasonal adjustment mode (multiplicative) are in conflict. Thus when specifying both the REGRESSION and X11 statements, it is necessary to also specify either a transform (using the TRANSFORM statement) or a mode (using the MODE= option of the X11 statement) in order to seasonally adjust the data using the regARIMA model.

According to Ladiray and Quenneville (2001), "X-12-ARIMA is based on the same principle [as the X-11 method] but proposes, in addition, a complete module, called

Reg-ARIMA, that allows for the initial series to be corrected for all sorts of undesirable effects. These effects are estimated using regression models with ARIMA errors (Findley et al. [23]).” In order to correct the series for effects in this manner, the REGRESSION statement must be specified. The effects which may be corrected in this manner are listed in the PREDEFINED= option below.

The following options can appear in the REGRESSION statement.

PREDEFINED= CONSTANT
PREDEFINED= LOM
PREDEFINED= LOMSTOCK
PREDEFINED= LOQ
PREDEFINED= LPYEAR
PREDEFINED= SEASONAL
PREDEFINED= TD
PREDEFINED= TDNOLPYEAR
PREDEFINED= TD1COEF
PREDEFINED= TD1NOLPYEAR

lists the predefined regression variables to be included in the model. Data values for these variables are calculated by the program, mostly as functions of the calendar. Table 32.2 gives definitions for the available predefined variables. The values LOM and LOQ are actually equivalent: the actual regression is controlled by the PROC X12 SEASONS= option. Multiple predefined regression variables can be used. The syntax for using both a length-of-month and a seasonal regression could be in one of the following forms:

```
regression predefined=lom seasonal;

regression predefined=(lom seasonal);

regression predefined=lom predefined=seasonal;
```

Certain restrictions apply when using more than one predefined regression variable. Only one of TD, TDNOLPYEAR, TD1COEF, or TD1NOLPYEAR may be specified. LPYEAR cannot be used with TD, TD1COEF, LOM, LOMSTOCK, or LOQ. LOM or LOQ cannot be used with TD or TD1COEF.

Table 32.2. Predefined Regression Variables in X-12-ARIMA

Regression Effect	**Variable Definitions**
Table 32.2. (continued)	
Trend Constant CONSTANT	$(1-B)^{-d}(1-B^s)^{-D}I(t \geq 1)$, where $I(t \geq 1) = \begin{cases} 1 & \text{for } t \geq 1 \\ 0 & \text{for } t < 1 \end{cases}$
Length-of-Month (monthly flow) LOM	$m_t - \bar{m}$ where m_t = length of month t (in days) and $\bar{m} = 30.4375$ (average length of month)
Stock Length-of-Month LOMSTOCK	$SLOM_t = \begin{cases} m_t - \bar{m} - \mu(l) & \text{for t = 1} \\ SLOM_{t-1} + m_t - \bar{m} & \text{otherwise} \end{cases}$ where $\bar{m}$ and m_t are defined in LOM and $\mu(l) = \begin{cases} 0.375 & \text{when 1st February in series is a leap year} \\ 0.125 & \text{when 2nd February in series is a leap year} \\ -0.125 & \text{when 3rd February in series is a leap year} \\ -0.375 & \text{when 4th February in series is a leap year} \end{cases}$
Length-of-Quarter (quarterly flow) LOQ	$q_t - \bar{q}$ where q_t = length of quarter t (in days) and $\bar{q} = 91.3125$ (average length of quarter)
Leap Year (monthly and quarterly flow) LPYEAR	$LY_t = \begin{cases} 0.75 & \text{in leap year February (first quarter)} \\ -0.25 & \text{in other Februaries (first quarter)} \\ 0 & \text{otherwise} \end{cases}$
Fixed Seasonal SEASONAL	$M_{1,t} = \begin{cases} 1 & \text{in January} \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases}$,..., $M_{11,t} = \begin{cases} 1 & \text{in November} \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases}$
Trading Day TD, TDNOLPYEAR	$T_{1,t} = (no.of Mondays) - (no.of Sundays)$, ..., $T_{6,t} = (no.of Saturdays) - (no.of Sundays)$
One Coefficient Trading Day TD1COEF, TD1NOLPYEAR	$(no.of weekdays) - \frac{5}{2}(no.of Saturdays and Sundays)$

TRANSFORM Statement

TRANSFORM *options;*

The TRANSFORM statement transforms or adjusts the series prior to estimating a regARIMA model. With this statement, the series can be Box-Cox (power) transformed. The Prior Adjustment Factors table is associated with the TRANSFORM statement.

Only one of the following options can appear in the TRANSFORM statement.

POWER= *value*

Transform the input series Y_t using a Box-Cox power transformation,

$$Y_t \rightarrow y_t = \begin{cases} log(Y_t) & \lambda = 0 \\ \lambda^2 + (Y_t^\lambda - 1)/\lambda & \lambda \neq 0 \end{cases}$$

The power λ must be specified (for example, POWER= .33). The default is no transformation ($\lambda = 1$); that is, POWER= 1. The log transformation (POWER= 0), square root transformation (POWER= .5), and the inverse transformation (POWER= -1) are equivalent to the corresponding FUNCTION= option.

Table 32.3. Power Values Related to the Census Bureau Function Argument

function=	**transformation**	**range for Y_t**	**equivalent power argument**
none	Y_t	all values	$power = 1$
log	$log(Y_t)$	$Y_t > 0$ for all t	$power = 0$
sqrt	$2(\sqrt{Y_t} - 0.875)$	$Y_t \geq 0$ for all t	$power = 0.5$
inverse	$2 - \frac{1}{Y_t}$	$Y_t \neq 0$ for all t	$power = -1$
logistic	$log(\frac{Y_t}{1-Y_t})$	$0 < Y_t < 1$ for all t	*none equivalent*

FUNCTION=NONE
FUNCTION=LOG
FUNCTION=SQRT
FUNCTION=INVERSE
FUNCTION=LOGISTIC
FUNCTION=AUTO

The transformation used by FUNCTION=NONE, LOG, SQRT, INVERSE, and LOGISTIC is related to the POWER= option as shown in Table 32.3. FUNCTION=AUTO uses an AIC-based selection to decide between a log transformation and no transformation. The default is FUNCTION=NONE.

However, the FUNCTION= and POWER= options are not completely equivalent. In some cases, using the FUNCTION= option causes the program to automatically select other options. For instance, FUNCTION=NONE causes the default mode to be MODE=ADD in the X11 statement. Also, the choice of transformation invoked by the FUNCTION=AUTO option may impact the default mode of the X11 statement.

Note that there are restrictions on the value used in the POWER and FUNCTION options when preadjustment factors for seasonal adjustment are generated from a regARIMA model. When seasonal adjustment is requested with the X11 statement, any value of the POWER option can be used for the purpose of forecasting the series with a regARIMA model. However, this is not the case when factors generated from the regression coefficients are used to adjust either the original series or the final seasonally adjusted series. In this case, the only accepted transformations are the log transformation, which can be specified as POWER = 0 (for multiplicative or log-additive seasonal adjustments) and no transformation, which can be specified as POWER = 1 (for additive seasonal adjustments). If no seasonal adjustment is performed, any POWER transformation can be used. The above restrictions also apply to FUNCTION=NONE and FUNCTION=LOG.

VAR Statement

VAR *variables;*

The VAR statement is used to specify the variables in the input data set that are to be analyzed by the procedure. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are analyzed except those appearing in a BY or ID statement or the variable named in the DATE= option in the PROC X12 statement.

X11 Statement

X11 *options;*

The X11 statement is an optional statement for invoking seasonal adjustment by an enhanced version of the methodology of the Census Bureau X-11 and X-11Q programs. You can control the type of seasonal adjustment decomposition calculated with the MODE= option. The output includes the final tables and diagnostics for the X-11 seasonal adjustment method listed below in Table 32.4.

Table 32.4. Tables Related to X11 Seasonal Adjustment

Table Name	Description
B1	original series, adjusted for prior effects and forecast extended
C17	final weights for the irregular component
C20	final extreme value adjustment factors
D1	modified original data, D iteration
D7	preliminary trend cycle, D iteration
D8	final unmodified SI ratios (differences)
D8A	*F* tests for stable and moving seasonality, D8
D9	final replacement values for extreme SI ratios (differences), D iteration
D9A	moving seasonality ratios for each period
D10	final seasonal factors
D10D	final seasonal difference
D11	final seasonally adjusted series
D11A	final seasonally adjusted series with forced yearly totals
D11R	rounded final seasonally adjusted series (with forced yearly totals)
D12	final trend-cycle
D13	final irregular component
D16	combined seasonal and trading day factors
D16B	final adjustment differences
D18	combined calendar adjustment factors
E4	ratio of yearly totals of original and seasonally adjusted series
E5	percent changes (differences) in original series
E6	percent changes (differences) in seasonally adjusted series
E6A	percent changes (differences) in seasonally adjusted series with forced yearly totals (D11.A)
E6R	percent changes (differences) in rounded seasonally adjusted series (D11.R)
E7	percent changes (differences) in final trend component series
F2A - F2I	X11 diagnostic summary
F3	monitoring and quality assessment statistics
F4	day of the week trading day component factors
G	spectral plots

For more details on the X-11 seasonal adjustment diagnostics, refer to Shiskin, Young, and Musgrave (1967), Lothian and Morry (1978), and Ladiray and Quenneville (2001).

The following options can appear in the X11 statement.

MODE= ADD
MODE= MULT
MODE= LOGADD
MODE= PSEUDOADD

determines the mode of the seasonal adjustment decomposition to be performed. There are four choices: multiplicative (MODE=MULT), additive (MODE=ADD), pseudo-additive (MODE=PSEUDOADD), and log-additive (MODE=LOGADD) decomposition. If this option is omitted, the procedure performs multiplicative adjustments. Table 32.5 shows the values of the MODE= option and the corresponding models for the original (O) and the seasonally adjusted (SA) series.

Table 32.5. Modes of Seasonal Adjustment and Their Models

Value of Mode Option	Name	Model for O	Model for SA
mult	Multiplicative	$O = C \times S \times I$	$SA = C \times I$
add	Additive	$O = C + S + I$	$SA = C + I$
pseudoadd	Pseudo-Additive	$O = C \times [S + I - 1]$	$SA = C \times I$
logadd	Log-Additive	$Log(O) = C + S + I$	$SA = exp(C + I)$

OUTFCST
OUTFORECAST

determines if forecasts will be included in certain tables sent to the output data set. If OUTFORECAST is specified, then forecasted values will be included in the output data set for tables A6, A8, A16, B1, D10, D10D, D16, D16B, and D18. The default is not to include forecasts.

SEASONALMA=S3X1
SEASONALMA=S3X3
SEASONALMA=S3X5
SEASONALMA=S3X9
SEASONALMA=S3X15
SEASONALMA=STABLE
SEASONALMA=X11DEFAULT
SEASONALMA=MSR

specifies which seasonal moving average (also called seasonal “filter”) will be used to estimate the seasonal factors. These seasonal moving averages are $n \times m$ *moving averages*, meaning that an n-term simple average is taken of a sequence of consecutive m-term simple averages. X11DEFAULT is the method used by the U.S. Census Bureau’s X-11-ARIMA program. The default for PROC X12 is SEASONALMA=MSR, which is the methodology of Statistic Canada’s X-11-ARIMA/88 program. Table 32.6 describes the seasonal filter options available.

The following seasonal filters can be selected for the entire series:

Table 32.6. X-12-ARIMA Seasonal Filter Options and Descriptions

Filter Name	Description of Filter
S3X1	A 3×1 moving average.
S3X3	A 3×3 moving average.
S3X5	A 3×5 moving average.
S3X9	A 3×9 moving average.
S3X15	A 3×15 moving average.
STABLE	Stable seasonal filter. As single seasonal factor for each calendar month or quarter is generated by calculating the simple average of all the values for each month or quarter (taken after detrending and outlier adjustment).
X11DEFAULT	A 3×3 moving average is used to calculate the initial seasonal factors in each iteration, and a 3×5 moving average to calculate the final seasonal factors.
MSR	Choose filter automatically using moving seasonality ratio of X-11-ARIMA/88 (Dagum 1988).

TRENDMA= *value*

specifies which Henderson moving average will be used to estimate the final trend-cycle. Any odd number greater than one and less than or equal to 101 can be specified. Example: trendma=23. If no selection is made the program will select a trend moving average based on statistical characteristics of the data. For monthly series, either a 9-, 13-, or 23-term Henderson moving average will be selected. For quarterly series, the program will choose either a 5- or a 7-term Henderson moving average.

FINAL= AO
FINAL= LS
FINAL= TC
FINAL= ALL

Lists the types of prior adjustment factors, obtained from the regression and outlier statements, that are to be removed from the final seasonally adjusted series. Additive outliers (FINAL=AO), level change and ramp outliers (FINAL=LS), and temporary change (FINAL=TC) can be removed. If this option is not specified, the final seasonally adjusted series will contain these effects.

FORCE= TOTALS
FORCE= ROUND
FORCE= BOTH

specifies that the seasonally adjusted series be modified to (a) force the yearly totals of the seasonally adjusted series and the original series be the same (FORCE=TOTALS), (b) adjust the seasonally adjusted values for each calendar year so that the sum of the rounded seasonally adjusted series for any year will equal the rounded annual total (FORCE=ROUND), or (c) first force the yearly totals, then round the adjusted series (FORCE=BOTH). When FORCE=TOTALS, the differences between the annual totals is distributed over the seasonally adjusted values in a way that approximately preserves the month-to-month (or quarter-to-quarter) movements of the original series. For more details refer to Huot (1975) and Cholette (1979). This forcing procedure is not recommended if the seasonal pattern is changing or if trading day adjustment is performed. Forcing the seasonally adjusted totals to be the same as the original series annual totals can degrade the quality of the seasonal adjustment, especially when the seasonal pattern is undergoing change. It is not natural if trading day adjustment is performed because the aggregate trading day effect over a year is variable and moderately different from zero.

Details

Missing Values

PROC X12 can process a series with missing values. Missing values in a series are considered to be one of two types. One type of missing value is a leading or trailing missing value; these values occur before the first non-missing value or after the last non-missing value in the span of a series. The span of a series may either be determined explicitly by the SPAN= option of the PROC X12 statement or implicitly by the START= or DATE= options. By default leading and trailing missing values are ignored. The second type of missing value is an embedded missing value; these missing values occur between the first non-missing value and the last non-missing value in the span of the series. Embedded missing values are processed using X-12-ARIMA's missing value method described below. The NOTRIMMISS option of the PROC X12 statement causes leading and trailing missing values to also be processed using the X-12-ARIMA missing value method.

When the X-12-ARIMA method encounters a missing value, it inserts an additive outlier for that observation into the set of regression variables for the model of the series and then replaces the missing observation with a value large enough to be considered an outlier during model estimation. After the regARIMA model is estimated, the X-12-ARIMA method adjusts the original series using factors generated from these missing value outlier regressors. The adjusted values are estimates of the missing values, and the adjusted series is displayed in Table MV1.

Computations

For more details on the computations used in PROC X12, refer to *X-12-ARIMA Reference Manual* (U.S. Bureau of the Census 2001b).

For more details on the X-11 method of decomposition, refer to *Seasonal Adjustment with the X-11 Method* (Ladiray and Quenneville 2001).

Displayed Output/ODS Table Names/OUTPUT Tablename Keywords

The options specified in PROC X12 control both the tables produced by the procedure and the tables available for output to the OUT= data set specified in the OUTPUT statement.

The displayed output is organized into tables identified by a part letter and a sequence number within the part. The seven major parts of the X12 procedure are as follows:

A	prior adjustments and regARIMA components (optional)
B	preliminary estimates of irregular component weights and trading-day regression factors (X-11 method)
C	final estimates of irregular component weights and trading-day regression factors
D	final estimates of seasonal, trend cycle, and irregular components
E	analytical tables
F	summary measures
G	charts

Table 32.7 describes the individual tables and charts. “P” indicates that the table is displayed only and is not available for output to the OUT= data set. Data from displayed tables can be extracted into data sets using the Output Delivery System (ODS). Refer to Chapter 6, “Using the Output Delivery System,” in the *SAS/ETS User’s Guide*. “O” indicates that the table is available only using the OUTPUT statement. The actual number of tables displayed depends on the options and statements specified. If a table is not computed, it is not displayed.

Table 32.7. Table Names and Descriptions

Table	**Description**	**Notes**
A1	original series	
A2	prior-adjustment factors	
RegParameterEstimates	regression model parameter estimates	P
ACF	autocorrelation factors	P
PACF	partial autocorrelation factors	P
ARMAIterationTolerances	exact ARMA likelihood estimation iteration tolerances	P
IterHistory	ARMA iteration history	P
ARMAIterationSummary	exact ARMA likelihood estimation iteration summary	P
RegressorGroupChiSq	chi-squared tests for groups of regressors	P
ARMAParameterEstimates	exact ARMA maximum likelihood estimation	P
AvgFcstErr	average absolute percentage error in within(out)-sample fore(back)casts:	P
Roots	(non)seasonal (AR)MA roots	P
MLESummary	estimation summary	P
ForecastCL	forecasts, standard errors, and confidence limits	P
MV1	original series adjusted for missing value regressors	
A6	regARIMA trading day component	
A8	regARIMA combined outlier component	
A8AO	regARIMA AO outlier component	
A8LS	regARIMA level change outlier component	
A8TC	regARIMA temporary change outlier component	
B1	prior adjusted or original series	
C17	final weight for irregular components	
C20	final extreme value adjustment factors	O
D1	modified original data, D iteration	O
D7	preliminary trend cycle, D iteration	O

Table 32.7. (continued)

Table	Description	Notes
D8	final unmodified S-I ratios	
D8A	seasonality tests	P
D9	final replacement values for extreme S-I ratios	
D9A	moving seasonality ratio	P
D10	final seasonal factors	
D10D	final seasonal difference	
D11	final seasonally adjusted series	
D11A	final seasonally adjusted series with forced yearly totals	
D11R	rounded final seasonally adjusted series (with forced yearly totals)	
D12	final trend cycle	
D13	final irregular series	
D16	combined adjustment factors	
D16B	final adjustment differences	
D18	combined calendar adjustment factors	
E4	ratios of annual totals	P
E5	percent changes in original series	
E6	percent changes in final seasonally adjusted series	
E6A	percent changes (differences) in seasonally adjusted series with forced yearly totals (D11.A)	
E6R	percent changes (differences) in rounded seasonally adjusted series (D11.R)	
E7	differences in final trend cycle	
F2A-I	summary measures	P
F3	quality assessment statistics	P
F4	Day of the Week Trading Day Component Factors	P
G	spectral analysis	P

ODS Graphics (Experimental)

This section describes the use of ODS for creating graphics with the X12 procedure. These graphics are experimental in this release, meaning that both the graphical results and the syntax for specifying them are subject to change in a future release.

To request these graphs, you must specify the ODS GRAPHICS statement. For more information on the ODS GRAPHICS statement, see Chapter 9, “Statistical Graphics Using ODS.”

The graphics available through ODS GRAPHICS are ACF plots, PACF plots, and spectral graphs. ACF and PACF plots are not available unless the IDENTIFY statement is used. A spectral plot of the original series is always available; however additional spectral plots are provided when the X11 statement is used. When the ODS GRAPHICS statement is not used, the plots are integrated into the ACF, PACF and spectral tables as a column of the table.

ODS Graph Names

PROC X12 assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 32.8.

Table 32.8. ODS Graphics Produced by PROC X12

ODS Graph Name	Plot Description
ACFPlot	Autocorrelation of Model Residuals
PACFPlot	Partial Autocorrelation of Model Residuals
SpectralPlot	Spectral Plot of Original or Adjusted Series

Examples

Example 32.1. Model Identification

An example of the statements typically invoked when using PROC X12 for model identification might follow the same format as the following example. This example invokes the X12 procedure and uses the TRANSFORM and IDENTIFY statements. It specifies the time series data, takes the logarithm of the series (TRANSFORM statement), and generates ACFs and PACFs for the specified levels of differencing (IDENTIFY statement). The ACFs and PACFs for Nonseasonal Order=1 and Seasonal Order=1 are shown in Output 32.1.1, Output 32.1.2, Output 32.1.3 and Output 32.1.4. The data set is the same as in the section "Basic Seasonal Adjustment" on page 1926.

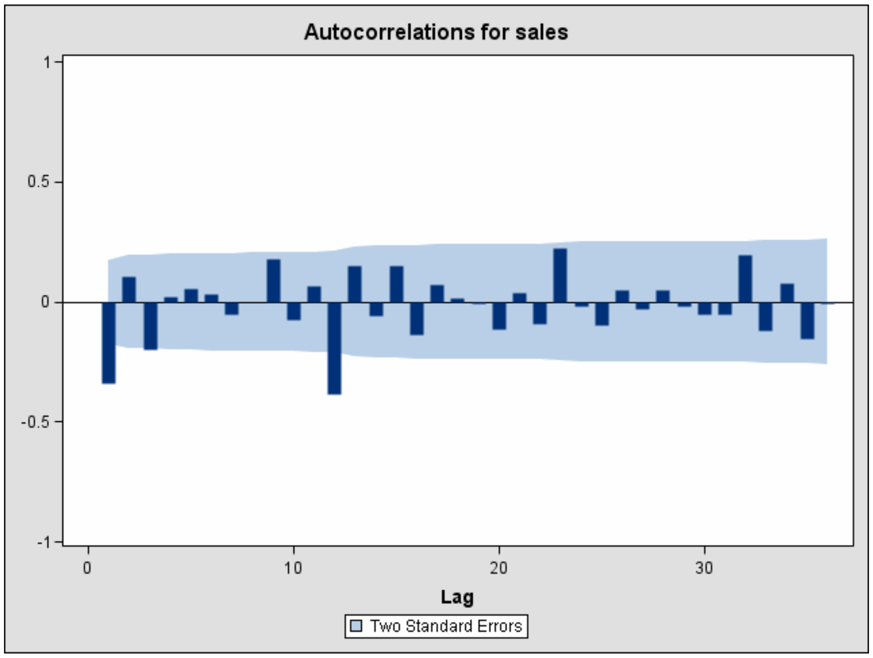
The graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the X12 procedure, see the "ODS Graphics" section on page 1951.

```
ods html;
ods graphics on;

proc x12 data=sales date=date;
   var sales;
   transform power=0;
   identify diff=(0,1) sdiff=(0,1);
run ;

ods graphics off;
ods html close;
```

Output 32.1.1. ACFs (Nonseasonal Order=1 Seasonal Order=1)

```
                              The X12 Procedure

                        Autocorrelation of Model Residuals
                Differencing:  Nonseasonal Order=1 Seasonal Order=1
                              For variable sales
                             Standard
      Lag    Correlation        Error    Chi-Square        DF    Pr > ChiSq

        1       -0.34112      0.08737       15.5957         1        <.0001
        2        0.10505      0.09701       17.0860         2        0.0002
        3       -0.20214      0.09787       22.6478         3        <.0001
        4        0.02136      0.10101       22.7104         4        0.0001
        5        0.05565      0.10104       23.1387         5        0.0003
        6        0.03080      0.10128       23.2709         6        0.0007
        7       -0.05558      0.10135       23.7050         7        0.0013
        8       -0.00076      0.10158       23.7050         8        0.0026
        9        0.17637      0.10158       28.1473         9        0.0009
       10       -0.07636      0.10389       28.9869        10        0.0013
       11        0.06438      0.10432       29.5887        11        0.0018
       12       -0.38661      0.10462       51.4728        12        <.0001
       13        0.15160      0.11501       54.8664        13        <.0001
       14       -0.05761      0.11653       55.3605        14        <.0001
       15        0.14957      0.11674       58.7204        15        <.0001
       16       -0.13894      0.11820       61.6452        16        <.0001
       17        0.07048      0.11944       62.4045        17        <.0001
       18        0.01563      0.11975       62.4421        18        <.0001
       19       -0.01061      0.11977       62.4596        19        <.0001
       20       -0.11673      0.11978       64.5984        20        <.0001
       21        0.03855      0.12064       64.8338        21        <.0001
       22       -0.09136      0.12074       66.1681        22        <.0001
       23        0.22327      0.12126       74.2099        23        <.0001
       24       -0.01842      0.12436       74.2652        24        <.0001
       25       -0.10029      0.12438       75.9183        25        <.0001
       26        0.04857      0.12500       76.3097        26        <.0001
       27       -0.03024      0.12514       76.4629        27        <.0001
       28        0.04713      0.12520       76.8387        28        <.0001
       29       -0.01803      0.12533       76.8943        29        <.0001
       30       -0.05107      0.12535       77.3442        30        <.0001
       31       -0.05377      0.12551       77.8478        31        <.0001
       32        0.19573      0.12569       84.5900        32        <.0001
       33       -0.12242      0.12799       87.2543        33        <.0001
       34        0.07775      0.12888       88.3401        34        <.0001
       35       -0.15245      0.12924       92.5584        35        <.0001
       36       -0.01000      0.13061       92.5767        36        <.0001

NOTE: The P-values approximate the probability of observing a Q-value at least
      this large when the model fitted is correct. When DF is positive, small
      values of P, customarily those below 0.05 indicate model inadequacy.
```

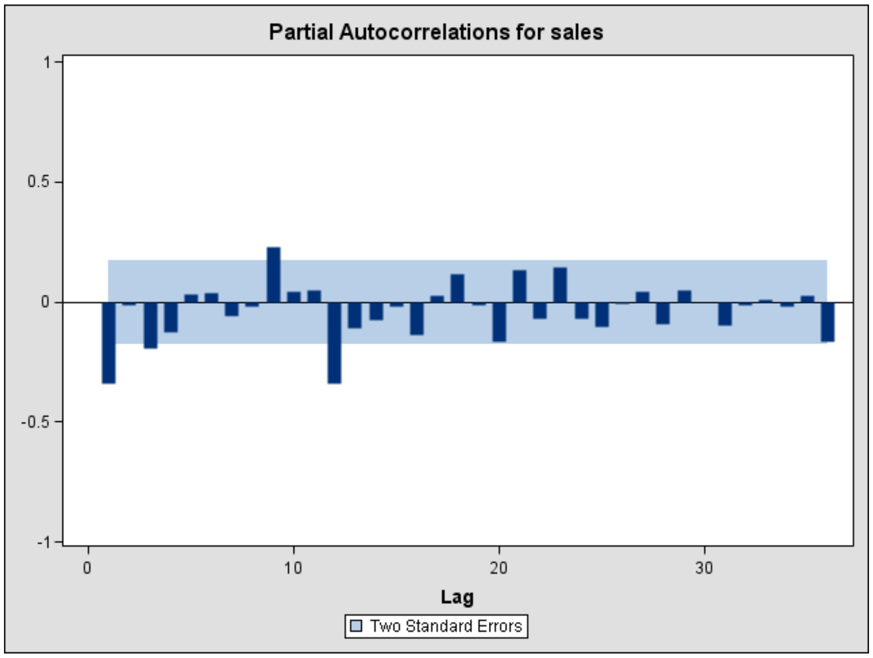
Output 32.1.2. Plot for ACFs (Nonseasonal Order=1 Seasonal Order=1) (Experimental)



Output 32.1.3. PACFs (Nonseasonal Order=1 Seasonal Order=1)

```
                              The X12 Procedure

                            Partial Autocorrelation
                              of Model Residuals
                          Differencing:  Nonseasonal
                           Order=1 Seasonal Order=1
                              For variable sales
                                                   Standard
                        Lag    Correlation            Error

                          1       -0.34112          0.08737
                          2       -0.01281          0.08737
                          3       -0.19266          0.08737
                          4       -0.12503          0.08737
                          5        0.03309          0.08737
                          6        0.03468          0.08737
                          7       -0.06019          0.08737
                          8       -0.02022          0.08737
                          9        0.22558          0.08737
                         10        0.04307          0.08737
                         11        0.04659          0.08737
                         12       -0.33869          0.08737
                         13       -0.10918          0.08737
                         14       -0.07684          0.08737
                         15       -0.02175          0.08737
                         16       -0.13955          0.08737
                         17        0.02589          0.08737
                         18        0.11482          0.08737
                         19       -0.01316          0.08737
                         20       -0.16743          0.08737
                         21        0.13240          0.08737
                         22       -0.07204          0.08737
                         23        0.14285          0.08737
                         24       -0.06733          0.08737
                         25       -0.10267          0.08737
                         26       -0.01007          0.08737
                         27        0.04378          0.08737
                         28       -0.08995          0.08737
                         29        0.04690          0.08737
                         30       -0.00490          0.08737
                         31       -0.09638          0.08737
                         32       -0.01528          0.08737
                         33        0.01150          0.08737
                         34       -0.01916          0.08737
                         35        0.02303          0.08737
                         36       -0.16488          0.08737
```

Output 32.1.4. Plot for PACFs (Nonseasonal Order=1 Seasonal Order=1) (Experimental)



Example 32.2. Model Estimation

After studying the output from Example 32.1 and identifying the ARIMA part of the model as, for example, (0 1 1)(0 1 1) 12, you can replace the IDENTIFY statement with the ARIMA and ESTIMATE statements. The parameter estimates and estimation summary statistics are shown in Output 32.2.1.

```
proc x12 data=sales date=date;
   var sales;
   transform power=0;
   arima model=( (0,1,1)(0,1,1) );
   estimate;
run ;
```

Output 32.2.1. Estimation Data

```
                              The X12 Procedure

                  Exact ARMA Likelihood Estimation Iteration Tolerances
                                For variable sales

                   Maximum Total ARMA Iterations           200
                   Convergence Tolerance               1.0E-05


               Average absolute percentage error in within-sample forecasts:
                              For variable sales

                           Last year:               2.81
                           Last-1 year:             6.38
                           Last-2 year:             7.69
                           Last three years:        5.63


                    Exact ARMA Likelihood Estimation Iteration Summary
                              For variable sales

                   Number of ARMA iterations                    6
                   Number of Function Evaluations              19


                         Exact ARMA Maximum Likelihood Estimation
                              For variable sales
                                                  Standard
Parameter                 Lag      Estimate          Error    t Value    Pr > |t|

Nonseasonal MA              1       0.40181        0.07887       5.09      <.0001
Seasonal MA                12       0.55695        0.07626       7.30      <.0001


                              Estimation Summary
                              For variable sales

                   Number of Residuals                        131
                   Number of Parameters Estimated               3
                   Variance Estimate                      1.3E-03
                   Standard Error Estimate                3.7E-02
                   Log likelihood                        244.6965
                   Transformation Adjustment            -735.2943
                   Adjusted Log likelihood              -490.5978
                   AIC                                   987.1956
                   AICC (F-corrected-AIC)                987.3845
                   Hannan Quinn                          990.7005
                   BIC                                   995.8211
```

Example 32.3. Seasonal Adjustment

Assuming that the model in Example 32.2 is satisfactory, a seasonal adjustment utilizing forecast extension can be performed by adding the X11 statement to the procedure. By default, the data is forecast one year ahead at the end of the series. Table D8.A is shown in Output 32.3.1.

```
ods output D8A#1=SalesD8A_1;
ods output D8A#2=SalesD8A_2;
ods output D8A#3=SalesD8A_3;
ods output D8A#4=SalesD8A_4;
```

```
proc x12 data=sales date=date;
   var sales;
   transform power=0;
   arima model=( (0,1,1)(0,1,1) );
   estimate;
   x11;
run ;



proc print data=SalesD8A_1;
   title 'Stable Seasonality Test';
run;

proc print data=SalesD8A_2;
   title 'Nonparametric Stable Seasonality Test';
run;

proc print data=SalesD8A_3;
   title 'Moving Seasonality Test';
run;

proc print data=SalesD8A_4;
   title 'Combined Seasonality Test';
run;
```

Output 32.3.1. Table D8.A as Displayed

```
                                The X12 Procedure

                      Table D 8.A:  F-tests for seasonality
                                For variable sales

             Test for the presence of seasonality assuming stability.
                          Sum of                    Mean
                         Squares          DF      Square     F-Value

   Between Months       23571.41          11    2142.855    190.9544    **
   Residual              1481.28         132    11.22182
   Total                25052.69         143

              ** Seasonality present at the 0.1 per cent level.


                        Nonparametric Test for the Presence
                         of Seasonality Assuming Stability
                        Kruskal-
                          Wallis                 Probability
                       Statistic          DF           Level

                        131.9546          11            .00%

                Seasonality present at the one percent level.


                             Moving Seasonality Test
                          Sum of                    Mean
                         Squares          DF      Square     F-Value

   Between Years        259.2517          10    25.92517    3.370317    **
   Error                846.1424         110    7.692204

            **Moving seasonality present at the one percent level.


           COMBINED TEST FOR THE PRESENCE OF IDENTIFIABLE SEASONALITY

                      IDENTIFIABLE SEASONALITY    PRESENT
```

The four ODS statements in the preceding example direct output from the D8A tables into four data sets: SalesD8A_1, SalesD8A_2, SalesD8A_3, and SalesD8A_4. It is best to direct the output to four different data sets because the four tables associated with table D8A have varying formats. The ODS data sets are shown below in Output 32.3.2.

Output 32.3.2. Table D8.A as Output in a Data Set Using ODS

```
                             Stable Seasonality Test

Obs    FT_SRC                  FT_SS      FT_DF       FT_MS       FT_F    FT_AST

 1     Between Months       23571.41         11    2142.855   190.9544     **
 2     Residual              1481.28        132    11.22182          .
 3     Total                25052.69        143           .          .



                      Nonparametric Stable Seasonality Test

                      Obs        KW_ST        KW_DF      KW_PR

                       1      131.9546           11      .00%



                             Moving Seasonality Test

Obs    FT_SRC                  FT_SS      FT_DF       FT_MS       FT_F    FT_AST

 1     Between Years        259.2517         10    25.92517   3.370317     **
 2     Error                846.1424        110    7.692204          .



                            Combined Seasonality Test

         Obs              Label1              cValue1         nValue1

          1     IDENTIFIABLE SEASONALITY    PRESENT                 .
```

Example 32.4. regARIMA Automatic Model Selection

This example demonstrates two of the new features available through the X-12-ARIMA method that are not available using the previous X-11 and X-11-ARIMA methods: regARIMA modeling and TRAMO-based automatic model selection. Assume that the same data set is used as in the previous examples.

```
ods select X12.ModelEstimation.AutoModel.UnitRootTestModel
           X12.ModelEstimation.AutoModel.UnitRootTest
           X12.ModelEstimation.AutoModel.AutoChoiceModel
           X12.ModelEstimation.AutoModel.Best5Model
           X12.ModelEstimation.AutoModel.AutomaticModelChoice
           X12.ModelEstimation.AutoModel.FinalModelChoice
           X12.ModelEstimation.AutoModel.AutomdlNote;
proc x12 data=sales date=date;
   var sales;
   transform function=log;
   regression predefined=td;
   automdl print=unitroottest unitroottestmdl autochoicemdl best5model;
   estimate;
   x11;
   output out=out a1 a2 a6 b1 c17 c20 d1 d7 d8 d9 d10
                  d11 d12 d13 d16 d18;
   run;
```

The Automatic Model Selection output is shown in Output 32.4.1 and Output 32.4.2. The first table in Output 32.4.1, "ARIMA Estimate for Unit Root Identification", gives details of the method that TRAMO uses to automatically select the orders of differencing. The second table, "Results of Unit Root Test for Identifying Orders of Differencing", shows that a regular difference order of 1 and a seasonal difference order of 1 has been determined by TRAMO. The third table, "Models estimated by Automatic ARIMA Model Selection procedure", shows all the models examined by the TRAMO-based method.

In Output 32.4.2, the first table, "Best Five ARIMA Models Chosen by Automatic Modeling", shows the top five models in order of rank and their BIC2 statistic. The second table, "Comparison of Automatically Selected Model and Default Model", compares the model selected by the TRAMO model to the default X-12-ARIMA model. The third table, "Final Automatic Model Selection", shows which model was actually selected.

Output 32.4.1. Output from the Automdl Statement

```
Automatic ARIMA Model Selection
Methodology based on research by Gomez and Maravall (1998).


                    ARIMA Estimates for Unit Root Identification
                                 For variable sales

         Model   Estimation                                ARMA
        Number       Method     Estimated Model    Parameter    Estimate

             1  H-R          ( 2, 0, 0)( 1, 0, 0)  NS_AR_1       0.67540
                H-R          ( 2, 0, 0)( 1, 0, 0)  NS_AR_2       0.28425
                H-R          ( 2, 0, 0)( 1, 0, 0)  S_AR_12       0.91963
             2  H-R          ( 1, 1, 1)( 1, 0, 1)  NS_AR_1       0.13418
                H-R          ( 1, 1, 1)( 1, 0, 1)  S_AR_12       0.98500
                H-R          ( 1, 1, 1)( 1, 0, 1)  NS_MA_1       0.47884
                H-R          ( 1, 1, 1)( 1, 0, 1)  S_MA_12       0.51726
             3  H-R          ( 1, 1, 1)( 1, 1, 1)  NS_AR_1      -0.39269
                H-R          ( 1, 1, 1)( 1, 1, 1)  S_AR_12       0.06223
                H-R          ( 1, 1, 1)( 1, 1, 1)  NS_MA_1      -0.09570
                H-R          ( 1, 1, 1)( 1, 1, 1)  S_MA_12       0.58536


                           Results of Unit Root Test for
                          Identifying Orders of Differencing
                                For variable sales

                          Regular     Seasonal
                       difference   difference          Mean
                            order        order   Significant

                                1            1         no


                Models estimated by Automatic ARIMA Model Selection procedure
                                For variable sales

       Model                                 ARMA              Statistics of Fit
       Number      Estimated Model    Parameter    Estimate         BIC       BIC2

           1    ( 3, 1, 0)( 0, 1, 0)  NS_AR_1      -0.33524
                ( 3, 1, 0)( 0, 1, 0)  NS_AR_2      -0.05558
                ( 3, 1, 0)( 0, 1, 0)  NS_AR_3      -0.15649
                ( 3, 1, 0)( 0, 1, 0)                          1024.469  -3.40549
           2    ( 3, 1, 0)( 0, 1, 1)  NS_AR_1      -0.33186
                ( 3, 1, 0)( 0, 1, 1)  NS_AR_2      -0.05823
                ( 3, 1, 0)( 0, 1, 1)  NS_AR_3      -0.15200
                ( 3, 1, 0)( 0, 1, 1)  S_MA_12       0.55279
                ( 3, 1, 0)( 0, 1, 1)                          993.7880  -3.63970

          ...output omitted...

          14    ( 0, 1, 1)( 0, 1, 0)  NS_MA_1       0.36005
                ( 0, 1, 1)( 0, 1, 0)                          1017.770  -3.45663
```

Output 32.4.2. Output from the Automdl Statement (Continued)

```
                         Best Five ARIMA Models Chosen
                             by Automatic Modeling
                              For variable sales

                     Rank     Estimated Model        BIC2

                        1  ( 0, 1, 1)( 0, 1, 1)  -3.69426
                        2  ( 1, 1, 0)( 0, 1, 1)  -3.69037
                        3  ( 1, 1, 1)( 0, 1, 1)  -3.65918
                        4  ( 0, 1, 2)( 0, 1, 1)  -3.65759
                        5  ( 2, 1, 0)( 0, 1, 1)  -3.65321


          Comparison of Automatically Selected Model and Default Model
                              For variable sales



                                                          Statistics of Fit
      Source of Candidate Models     Estimated Model       Plbox       Rvr

   Automatic Model Choice         ( 0, 1, 1)( 0, 1, 1)   0.62560   0.03546
   Airline Model (Default)        ( 0, 1, 1)( 0, 1, 1)   0.62561   0.03546

          Comparison of Automatically Selected Model and Default Model
                              For variable sales

                                                          Statistics of Fit
                                                                       Number
                                                                           of
   Source of Candidate Models     Estimated Model       Plbox       RvrOutliers

Automatic Model Choice         ( 0, 1, 1)( 0, 1, 1)   0.62560   0.03546       0
Airline Model (Default)        ( 0, 1, 1)( 0, 1, 1)   0.62561   0.03546       0

Check for Unit Roots

Check for Nonseasonal Overdifferencing

Check for for insignificant ARMA coefficients


                        Final Automatic Model Selection
                             For variable sales

                    Source of Model            Estimated Model

                 Automatic Model Choice         ( 0, 1, 1)( 0, 1, 1)
```

Table 32.9 and Output 32.4.3 illustrate the regARIMA modeling method. Table 32.9 shows the relationship between the output variables in PROC X12 that results from a regARIMA model. Note that some of these formulas apply only to this example. Output 32.4.3 shows the values of these variables for the first 24 observations in the example.

Table 32.9. regARIMA Output Variables and Descriptions

Table	Title	Type	Formula
A 1	Time series data (for the span analyzed)	data	Input.
A 2	Prior-adjustment factors Leap year (from trading day regression) adjustments.	factor	Calculated from regression.
A 6	RegARIMA trading day component Leap year prior adjustments included from Table A2.	factor	Calculated from regression.
B 1	Original series (prior adjusted) (adjusted for regARIMA factors)	data	$B1 = A1/A6$ * * because only TD specified.
C 17	Final weights for irregular component	factor	Calculated using moving standard deviation.
C 20	Final extreme value adjustment factors	factor	Calculated using C16 and C17.
D 1	Modified original data, D iteration	data	$D1 = B1/C20$ ** $D1 = C19/C20$ ** C19=B1 in this example.
D 7	Preliminary trend cycle, D iteration	data	Calculated using Henderson moving average.
D 8	Final unmodified SI ratios	factor	$D8 = B1/D7$ *** $D8 = C19/D7$ *** TD specified in regression.
D 9	Final replacement values for SI ratios	factor	If C17 shows extreme values, $D9 = D1/D7$; $D9 =$. otherwise.
D 10	Final seasonal factors	factor	Calculated using moving averages.
D 11	Final seasonally adjusted data (also adjusted for trading day)	data	$D11 = B1/D10$ **** $D11 = C19/D10$ **** $B1 = C19$ for this example.
D 12	Final trend cycle	data	Calculated using Henderson moving average.
D 13	Final irregular component	factor	$D13 = D11/D12$
D 16	Combined adjustment factors (includes seasonal, trading day factors)	factor	$D16 = A1/D11$
D 18	Combined calendar adjustment factors (includes trading day factors)	factor	$D18 = D16/D10$ $D18 = A6$ ***** ***** regression TD is the only calendar adjustment factor in this example.

Output 32.4.3. Output Variables Related to Trading-Day Regression

```
                Output Variables Related to Trading-Day Regression

                                                sales_  sales_
Obs date  sales_A1 sales_A2 sales_A6 sales_B1   C17     C20   sales_D1 sales_D7

  1 SEP78    112    1.00000  1.01328  110.532 1.00000 1.00000  110.532  124.138
  2 OCT78    118    1.00000  0.99727  118.323 1.00000 1.00000  118.323  124.905
  3 NOV78    132    1.00000  0.98960  133.388 1.00000 1.00000  133.388  125.646
  4 DEC78    129    1.00000  1.00957  127.777 1.00000 1.00000  127.777  126.231
  5 JAN79    121    1.00000  0.99408  121.721 1.00000 1.00000  121.721  126.557
  6 FEB79    135    0.99115  0.99115  136.205 1.00000 1.00000  136.205  126.678
  7 MAR79    148    1.00000  1.00966  146.584 1.00000 1.00000  146.584  126.825
  8 APR79    148    1.00000  0.99279  149.075 1.00000 1.00000  149.075  127.038
  9 MAY79    136    1.00000  0.99406  136.813 1.00000 1.00000  136.813  127.433
 10 JUN79    119    1.00000  1.01328  117.440 1.00000 1.00000  117.440  127.900
 11 JUL79    104    1.00000  0.99727  104.285 1.00000 1.00000  104.285  128.499
 12 AUG79    118    1.00000  0.99678  118.381 1.00000 1.00000  118.381  129.253
 13 SEP79    115    1.00000  1.00229  114.737 0.98631 0.99964  114.778  130.160
 14 OCT79    126    1.00000  0.99408  126.751 0.88092 1.00320  126.346  131.238
 15 NOV79    141    1.00000  1.00366  140.486 1.00000 1.00000  140.486  132.699
 16 DEC79    135    1.00000  0.99872  135.173 1.00000 1.00000  135.173  134.595
 17 JAN80    125    1.00000  0.99406  125.747 0.00000 0.95084  132.248  136.820
 18 FEB80    149    1.02655  1.03400  144.100 1.00000 1.00000  144.100  139.215
 19 MAR80    170    1.00000  0.99872  170.217 1.00000 1.00000  170.217  141.559
 20 APR80    170    1.00000  0.99763  170.404 1.00000 1.00000  170.404  143.777
 21 MAY80    158    1.00000  1.00966  156.489 1.00000 1.00000  156.489  145.925
 22 JUN80    133    1.00000  0.99279  133.966 1.00000 1.00000  133.966  148.133
 23 JUL80    114    1.00000  0.99406  114.681 0.00000 0.94057  121.927  150.682
 24 AUG80    140    1.00000  1.00957  138.673 1.00000 1.00000  138.673  153.774

                             sales_   sales_   sales_   sales_   sales_   sales_
Obs sales_D8  sales_D9    D10      D11      D12      D13      D16      D18

  1  0.89040     .      0.90264  122.453  124.448  0.98398  0.91463  1.01328
  2  0.94731     .      0.94328  125.438  125.115  1.00258  0.94070  0.99727
  3  1.06161     .      1.06320  125.459  125.723  0.99790  1.05214  0.98960
  4  1.01225     .      0.99534  128.375  126.205  1.01720  1.00487  1.00957
  5  0.96179     .      0.97312  125.083  126.479  0.98896  0.96735  0.99408
  6  1.07521     .      1.05931  128.579  126.587  1.01574  1.04994  0.99115
  7  1.15580     .      1.17842  124.391  126.723  0.98160  1.18980  1.00966
  8  1.17347     .      1.18283  126.033  126.902  0.99315  1.17430  0.99279
  9  1.07360     .      1.06125  128.916  127.257  1.01303  1.05495  0.99406
 10  0.91822     .      0.91663  128.121  127.747  1.00293  0.92881  1.01328
 11  0.81156     .      0.81329  128.226  128.421  0.99848  0.81107  0.99727
 12  0.91589     .      0.91135  129.897  129.316  1.00449  0.90841  0.99678
 13  0.88151  0.88182   0.90514  126.761  130.347  0.97249  0.90722  1.00229
 14  0.96581  0.96273   0.93820  135.100  131.507  1.02732  0.93264  0.99408
 15  1.05869     .      1.06183  132.306  132.937  0.99525  1.06571  1.00366
 16  1.00429     .      0.99339  136.072  134.720  1.01004  0.99212  0.99872
 17  0.91906  0.96658   0.97481  128.996  136.763  0.94321  0.96902  0.99406
 18  1.03509     .      1.06153  135.748  138.996  0.97663  1.09762  1.03400
 19  1.20245     .      1.17965  144.295  141.221  1.02177  1.17814  0.99872
 20  1.18520     .      1.18499  143.802  143.397  1.00283  1.18218  0.99763
 21  1.07239     .      1.06005  147.624  145.591  1.01397  1.07028  1.00966
 22  0.90436     .      0.91971  145.662  147.968  0.98442  0.91307  0.99279
 23  0.76108  0.80917   0.81275  141.103  150.771  0.93588  0.80792  0.99406
 24  0.90180     .      0.91133  152.166  154.161  0.98706  0.92005  1.00957
 ...output observations omitted...
```

Example 32.5. Automatic Outlier Detection

This example demonstrates using the OUTLIER statement to automatically detect and remove outliers from a time series to be seasonally adjusted. The data set is the same as in the section "Basic Seasonal Adjustment" on page 1926 and the previous examples. Adding the OUTLIER statement to Example 32.3 requests that outliers be detected using the default critical value as described in the section "OUTLIER Statement" on page 1939. The tables associated with outlier detection for this example are shown in Output 32.5.1. The first table shows the critical values; the second table shows that a single potential outlier was identified; the third table shows the estimates for the ARMA parameters. Since no outliers are included in the regression model, the Regression Model Parameter Estimates table is not displayed. Because only a potential outlier was identified, and not an actual outlier, in this case the A1 series and the B1 series are identical.

```
proc x12 data=sales date=date;
   var sales;
   transform function=log;
   arima model=( (0,1,1)(0,1,1) );
   outlier;
   estimate;
   x11;
   output out=nooutlier a1 b1 d10;
   run ;
```

Output 32.5.1. Proc X12 Output when Potential Outliers are Identified

```
                              The X12 Procedure

                   Critical Values to use in Outlier Detection
                             For variable sales

                         Begin                   SEP1978
                         End                     AUG1990
                         Observations                144
                         Method                  Add One
                         AO Critical Value      3.889838
                         LS Critical Value      3.889838


 NOTE:  The following time series values might later be identified as outliers
 when data are added or revised.  They were not identified as outliers in this
  run either because their test t-statistics were slightly below the critical
value or because they were eliminated during the backward deletion step of the
        identification procedure, when a non-robust t-statistic is used.

                             Potential Outliers
                             For variable sales
                   Type of                 t Value    t Value
                   Outlier    Date          for AO     for LS

                    AO        NOV1989        -3.48      -1.51


                     Exact ARMA Maximum Likelihood Estimation
                              For variable sales
                                                  Standard
Parameter                  Lag      Estimate         Error    t Value    Pr > |t|

Nonseasonal MA               1       0.40181       0.07887       5.09      <.0001
Seasonal MA                 12       0.55695       0.07626       7.30      <.0001
```

In the next example, reducing the critical value to 3.3 causes the outlier identification routine to more agressively identify outliers as shown in Output 32.5.2. The first table shows the critical values. The second table shows that three additive outliers and a level shift have been included in the regression model. The third table shows how the inclusion of outliers in the model affects the ARMA parameters.

```
proc x12 data=sales date=date;
   var sales;
   transform function=log;
   arima model=((0,1,1) (0,1,1));
   outlier cv=3.3;
   estimate;
   x11;
   output out=outlier a1 a8  a8ao a8ls b1 d10;
run;
```

Output 32.5.2. Proc X12 Output when Outliers are Identified

```
                                 The X12 Procedure

                    Critical Values to use in Outlier Detection
                                For variable sales

                          Begin                     SEP1978
                          End                       AUG1990
                          Observations                  144
                          Method                    Add One
                          AO Critical Value             3.3
                          LS Critical Value             3.3


                       Regression Model Parameter Estimates
                                For variable sales
                                                            Standard
Type                  Parameter             Estimate           Error  t Value  Pr > |t|

Automatically         AO JAN1981             0.09590         0.02168     4.42    <.0001
Identified
                      LS FEB1983            -0.09673         0.02488    -3.89    0.0002
                      AO OCT1983            -0.08032         0.02146    -3.74    0.0003
                      AO NOV1989            -0.10323         0.02480    -4.16    <.0001


                      Exact ARMA Maximum Likelihood Estimation
                                 For variable sales
                                                         Standard
Parameter                   Lag       Estimate              Error     t Value     Pr > |t|

Nonseasonal MA                1        0.33205            0.08239        4.03       <.0001
Seasonal MA                  12        0.49647            0.07676        6.47       <.0001
```

The first 65 observations of the A1, A8, A8AO, A8LS, B1 and D10 series are displayed in Output 32.5.3. The user can confirm the relationships,

$$A8 = A8AO \times A8LS$$

$$B1 = A1/A8$$

The seasonal factors are in series D10.

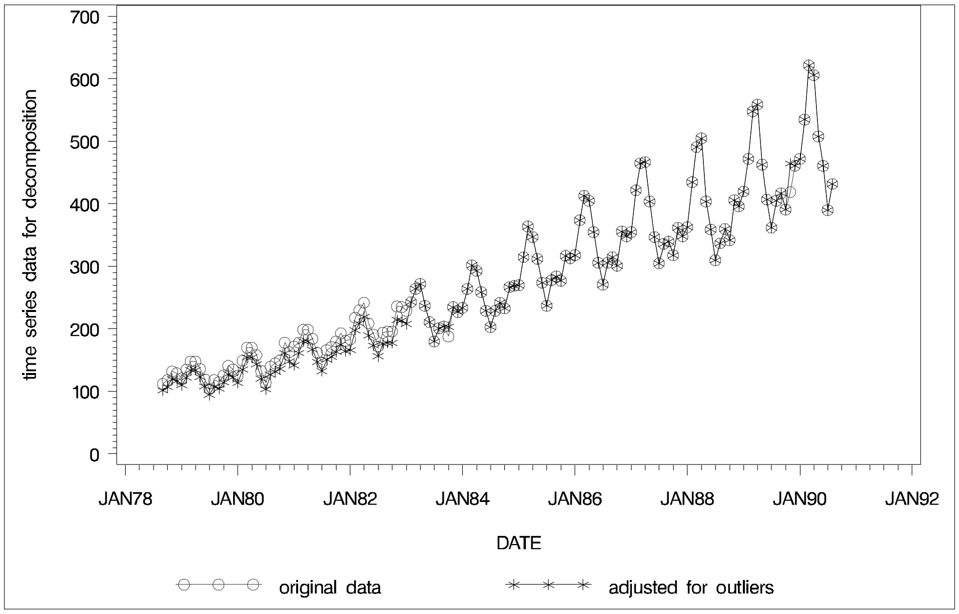
Output 32.5.3. Proc X12 Output Series Related to Outlier Detection

```
                                           sales_    sales_               sales_
 Obs    DATE     sales_A1    sales_A8       A8AO      A8LS     sales_B1      D10

   1    SEP78       112       1.10156    1.00000   1.10156    101.674    0.90496
   2    OCT78       118       1.10156    1.00000   1.10156    107.121    0.94487
   3    NOV78       132       1.10156    1.00000   1.10156    119.830    1.04711
   4    DEC78       129       1.10156    1.00000   1.10156    117.107    1.00119
   5    JAN79       121       1.10156    1.00000   1.10156    109.844    0.94833
   6    FEB79       135       1.10156    1.00000   1.10156    122.553    1.06817
   7    MAR79       148       1.10156    1.00000   1.10156    134.355    1.18679
   8    APR79       148       1.10156    1.00000   1.10156    134.355    1.17607
   9    MAY79       136       1.10156    1.00000   1.10156    123.461    1.07565
  10    JUN79       119       1.10156    1.00000   1.10156    108.029    0.91844
  11    JUL79       104       1.10156    1.00000   1.10156     94.412    0.81206
  12    AUG79       118       1.10156    1.00000   1.10156    107.121    0.91602
  13    SEP79       115       1.10156    1.00000   1.10156    104.397    0.90865
  14    OCT79       126       1.10156    1.00000   1.10156    114.383    0.94131
  15    NOV79       141       1.10156    1.00000   1.10156    128.000    1.04496
  16    DEC79       135       1.10156    1.00000   1.10156    122.553    0.99766
  17    JAN80       125       1.10156    1.00000   1.10156    113.475    0.94942
  18    FEB80       149       1.10156    1.00000   1.10156    135.263    1.07172
  19    MAR80       170       1.10156    1.00000   1.10156    154.327    1.18663
  20    APR80       170       1.10156    1.00000   1.10156    154.327    1.18105
  21    MAY80       158       1.10156    1.00000   1.10156    143.433    1.07383
  22    JUN80       133       1.10156    1.00000   1.10156    120.738    0.91930
  23    JUL80       114       1.10156    1.00000   1.10156    103.490    0.81385
  24    AUG80       140       1.10156    1.00000   1.10156    127.093    0.91466
  25    SEP80       145       1.10156    1.00000   1.10156    131.632    0.91302
  26    OCT80       150       1.10156    1.00000   1.10156    136.171    0.93086
  27    NOV80       178       1.10156    1.00000   1.10156    161.589    1.03965
  28    DEC80       163       1.10156    1.00000   1.10156    147.972    0.99440
  29    JAN81       172       1.21243    1.10065   1.10156    141.864    0.95136
  30    FEB81       178       1.10156    1.00000   1.10156    161.589    1.07981
  31    MAR81       199       1.10156    1.00000   1.10156    180.653    1.18661
  32    APR81       199       1.10156    1.00000   1.10156    180.653    1.19097
  33    MAY81       184       1.10156    1.00000   1.10156    167.036    1.06905
  34    JUN81       162       1.10156    1.00000   1.10156    147.064    0.92446
  35    JUL81       146       1.10156    1.00000   1.10156    132.539    0.81517
  36    AUG81       166       1.10156    1.00000   1.10156    150.695    0.91148
  37    SEP81       171       1.10156    1.00000   1.10156    155.234    0.91352
  38    OCT81       180       1.10156    1.00000   1.10156    163.405    0.91632
  39    NOV81       193       1.10156    1.00000   1.10156    175.206    1.03194
  40    DEC81       181       1.10156    1.00000   1.10156    164.312    0.98879
  41    JAN82       183       1.10156    1.00000   1.10156    166.128    0.95699
  42    FEB82       218       1.10156    1.00000   1.10156    197.901    1.09125
  43    MAR82       230       1.10156    1.00000   1.10156    208.795    1.19059
  44    APR82       242       1.10156    1.00000   1.10156    219.688    1.20448
  45    MAY82       209       1.10156    1.00000   1.10156    189.731    1.06355
  46    JUN82       191       1.10156    1.00000   1.10156    173.391    0.92897
  47    JUL82       172       1.10156    1.00000   1.10156    156.142    0.81476
  48    AUG82       194       1.10156    1.00000   1.10156    176.114    0.90667
  49    SEP82       196       1.10156    1.00000   1.10156    177.930    0.91200
  50    OCT82       196       1.10156    1.00000   1.10156    177.930    0.89970
  51    NOV82       236       1.10156    1.00000   1.10156    214.242    1.02160
  52    DEC82       235       1.10156    1.00000   1.10156    213.334    0.98632
  53    JAN83       229       1.10156    1.00000   1.10156    207.887    0.96343
  54    FEB83       243       1.00000    1.00000   1.00000    243.000    1.10282
  55    MAR83       264       1.00000    1.00000   1.00000    264.000    1.20199
  56    APR83       272       1.00000    1.00000   1.00000    272.000    1.21192
  57    MAY83       237       1.00000    1.00000   1.00000    237.000    1.06082
  58    JUN83       211       1.00000    1.00000   1.00000    211.000    0.93267
  59    JUL83       180       1.00000    1.00000   1.00000    180.000    0.81141
  60    AUG83       201       1.00000    1.00000   1.00000    201.000    0.90190
  61    SEP83       204       1.00000    1.00000   1.00000    204.000    0.91035
  62    OCT83       188       0.92283    0.92283   1.00000    203.722    0.88785
  63    NOV83       235       1.00000    1.00000   1.00000    235.000    1.00969
  64    DEC83       227       1.00000    1.00000   1.00000    227.000    0.97974
  65    JAN84       234       1.00000    1.00000   1.00000    234.000    0.97073
  . . . output observations omitted . . .
```

From the two previous examples, you can examine how outlier detection affects the seasonally adjusted series. Output 32.5.4 shows a plot of A1 versus B1 in the series where outliers are detected. B1 has been adjusted for the additive outliers and the level shift. Output 32.5.5 compares the seasonal factors (table D10) of the series unadjusted for outliers to the series adjusted for outliers. The seasonal factors are based on the B1 series.

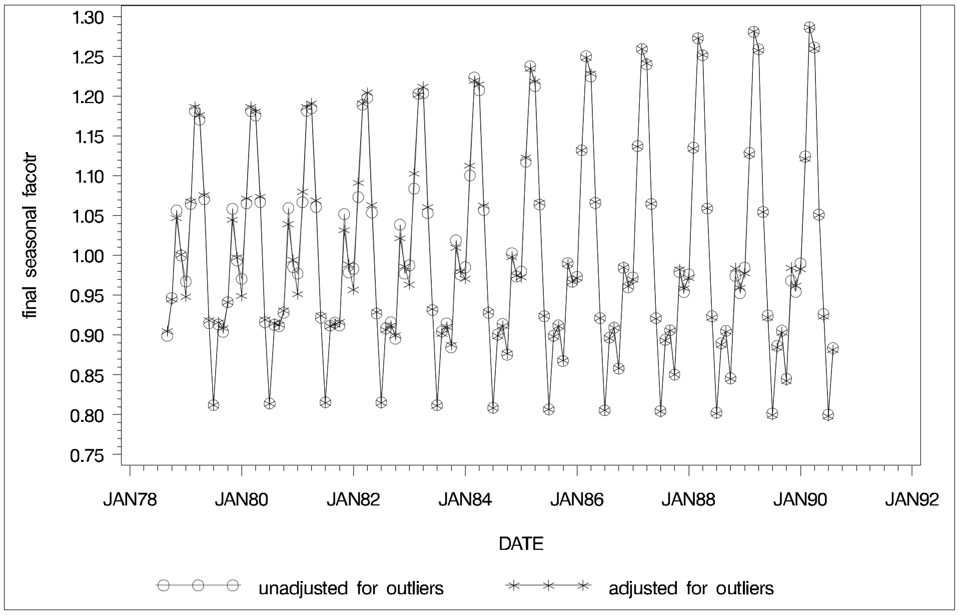
```
axis2 label=(angle=90  'time series data for decomposition');
symbol1 i=join v='star' c=black;
symbol2 i=join v='circle' c=red;
legend1 label=none value=('original data' 'adjusted for outliers');

proc gplot data=outlier;
      plot sales_A1    * date = 2
           sales_B1    * date = 1 / overlay legend=legend1 vaxis=axis2;
run;
```

Output 32.5.4. Original Series and Outlier Adjusted Series



```
data both;
  merge nooutlier(rename=(sales_D10=unadj_D10)) outlier;
run;

axis2 label=(angle=90  'final seasonal facotr');
symbol1 i=join v='star' c=blue;
symbol2 i=join v='circle' c=red;
legend1 label=none value=('unadjusted for outliers' 'adjusted for outliers');

proc gplot data=both;
      plot unadj_D10    * date = 2
           sales_D10    * date = 1 / overlay legend=legend1 vaxis=axis2;
run;
```

Output 32.5.5. Seasonal Factors Based on Original and Outlier Adjusted Series



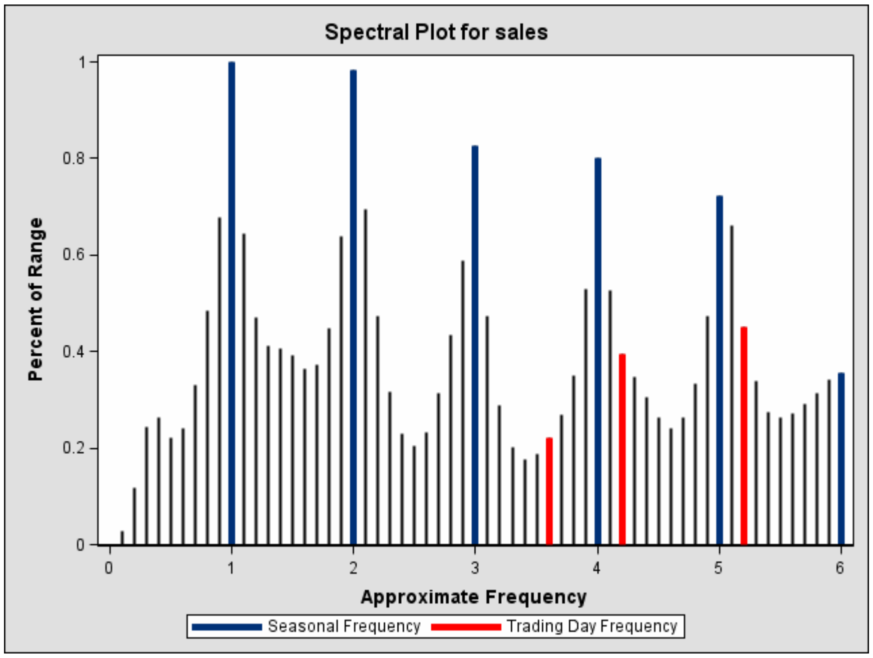
Example 32.6. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics. Using the same data set as in the section "Basic Seasonal Adjustment" on page 1926 and the previous examples, a Spectral Plot of the original series is displayed in Output 32.6.1.

The graphical displays are requested by specifying the experimental ODS GRAPHICS statement. For general information about ODS graphics, see Chapter 9, "Statistical Graphics Using ODS." For specific information about the graphics available in the X12 procedure, see the "ODS Graphics" section on page 1951.

```
ods html;
ods graphics on;

proc x12 data=sales date=date;
   var sales;
run ;

ods graphics off;
ods html close;
```

Output 32.6.1. Spectral Plot for Original Data (Experimental)



Acknowledgments

The X-12-ARIMA procedure was developed by the Time Series Staff of the Statistical Research Division, U.S. Bureau of the Census.

Brian Monsell is the primary programmer for the U.S. Census Bureau's X-12-ARIMA procedure and has been very helpful in the development of PROC X12.

The version of PROC X12 documented here was produced by converting the U.S. Census Bureau's FORTRAN code to the SAS development language and adding typical SAS procedure syntax. This conversion work was performed by SAS Institute, which resulted in the X12 procedure. Although several features were added during the conversion, credit for the statistical aspects and general methodology of the X12 procedure belongs to the U.S. Bureau of the Census.

The X-12-ARIMA seasonal adjustment program contains components developed from Statistics Canada's X-11-ARIMA program.

The X-12-ARIMA automatic modeling method is based on the work of Gomez and Maravall (1997a;b).

References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, Third Edition, Englewood Cliffs, NJ: Prentice Hall, Inc.

Cholette, P. A. (1979), “A Comparison and Assessment of Various Adjustment Methods of Sub-Annual Series to Yearly Benchmarks,” StatCan Staff Paper STC2119, Seasonal Adjustment and Time Series Staff, Statistics Canada, Ottawa.

Dagum, E. B. (1988), *The X-11-ARIMA/88 Seasonal Adjustment Method: Foundations and User’s Manual*, Ottawa: Statistics Canada.

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., and Chen, B. C. (1998), “New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program,” *Journal of Business and Economic Statistics*, 16, 127–176.

Gomez, V. and Maravall, A. (1997a), *Guide for Using the Programs TRAMO and SEATS, Beta Version*, Banco de Espana.

Gomez, V. and Maravall, A. (1997b), *Program TRAMO and SEATS: Instructions for the User, Beta Version*, Banco de Espana.

Huot, G. (1975), “Quadratic Minimization Adjustment of Monthly or Quarterly Series to Annual Totals,” StatCan Staff Paper STC2104, Statistics Canada, Seasonal Adjustment and Time Series Staff, Ottawa.

Ladiray, D. and Quenneville, B. (2001), *Seasonal Adjustment with the X-11 Method*,, New York: Springer-Verlag.

Ljung, G. M. (1993), “On Outlier Detection in Time Series,” *Journal of the Royal Statistical Society, B*, 55, 559–567.

Lothian, J. and Morry, M. (1978), “A Set of Quality Control Statistics for the X-11-ARIMA Seasonal Adjustment Method,” StatCan Staff Paper STC1788E, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Shiskin, J., Young, A. H., and Musgrave, J. C. (1967), “The X-11 Variant of the Census Method II Seasonal Adjustment Program,” Technical Report 15, US Department of Commerce, Bureau of the Census.

U.S. Bureau of the Census (2001a), *X-12-ARIMA Quick Reference for Unix, Version 0.2.8*, Washington, DC.

U.S. Bureau of the Census (2001b), *X-12-ARIMA Reference Manual, Version 0.2.8*, Washington, DC.

U.S. Bureau of the Census (2001c), *X-12-ARIMA Seasonal Adjustment Program, Version 0.2.8*, Washington, DC.

Part 3
Time Series Forecasting System

Contents

Chapter 33
Overview

Chapter Contents

Chapter 33
Overview of the Time Series Forecasting System

Introduction

The Time Series Forecasting system forecasts future values of time series variables by extrapolating trends and patterns in the past values of the series or by extrapolating the effect of other variables on the series. The system provides convenient point-and-click windows to drive the time series analysis and forecasting tools of SAS/ETS software.

You can use the system in a fully automatic mode, or you can use the system's diagnostic features and time series modeling tools interactively to develop forecasting models customized to best predict your time series. The system provides both graphical and statistical features to help you choose the best forecasting method for each series.

The following is a brief summary of the features of the Time Series Forecasting system. With the system you can

- use a wide variety of forecasting methods, including several kinds of exponential smoothing models, Winters method, and ARIMA (Box-Jenkins) models. You can also produce forecasts by combining the forecasts from several models.
- use predictor variables in forecasting models. Forecasting models can include time trend curves, regressors, intervention effects (dummy variables), adjustments you specify, and dynamic regression (transfer function) models.
- view plots of the data, predicted versus actual values, prediction errors, and forecasts with confidence limits, as well as autocorrelations and results of white noise and stationarity tests. Any of these plots can be zoomed and can represent raw or transformed series.
- use hold-out samples to select the best forecasting method
- compare goodness-of-fit measures for any two forecasting models side by side or list all models sorted by a particular fit statistic
- view the predictions and errors for each model in a spreadsheet or compare the fit of any two models in a spreadsheet
- examine the fitted parameters of each forecasting model and their statistical significance
- control the automatic model selection process: the set of forecasting models considered, the goodness-of-fit measure used to select the best model, and the time period used to fit and evaluate models

- customize the system by adding forecasting models for the automatic model selection process and for point-and-click manual selection
- save your work in a project catalog
- print an audit trail of the forecasting process
- show source statements for PROC ARIMA code
- save and print system output including spreadsheets and graphs

Using The Time Series Forecasting System

Chapters starting from Chapter 34 through Chapter 38 contain a series of example sessions that show the major features of the system. Chapters from Chapter 39 through Chapter 41 serve as reference and provide more details on how the system operates. The reference chapters contain a complete list of Forecasting system features.

To get started using the Time Series Forecasting system, it is a good idea to work through a few of the example sessions. Start with Chapter 34, "Getting Started with Time Series Forecasting," and use the system to reproduce the steps shown in the examples. Continue with the other chapters when you feel comfortable using the system.

The example sessions make use of time series data sets contained in the SASHELP library: `air, citimon, citiqtr, citiyr, citiwk, citiday, gnp, retail, usecon`, and `workers`. You can use these data sets to work through the example sessions or to experiment further with the Forecasting system.

Once you are familiar with how the system operates, start working with your own data to build your own forecasting models. When you have questions, consult the reference chapters mentioned above for more information about particular features.

The Time Series Forecasting system forecasts *time series*, that is, variables comprised of ordered observations taken at regular intervals over time. Since the Forecasting system is a part of the SAS software system, time series values must be stored as variables in a SAS data set or data view, with the observations representing the time periods. The data may also be stored in an external spreadsheet or data base if you license SAS/ACCESS software.

The Time Series Forecasting System chapters refer to *series* and *variables*. Since time series are stored as variables in SAS data sets or data views, these terms are used interchangeably. However, the term *series* is preferred when attention is focused on the sequence of data values, and the term *variable* is preferred when attention is focused on the data set.

Information Available on the World Wide Web

Visit the Time Series Forecasting Web site, support.sas.com/rnd/app/ets/forecasting.html, for up-to-date supplemental information. While using the forecasting system, click or select *SAS on the Web* and *Time Series Forecasting System* under the *Help* menu. The site contains:

- an overview of the Time Series Forecasting System
- information and examples on running the system in batch
- information and examples on creating forecasting reports for the Web
- documentation for the *Forecast Command Builder*, a utility for creating, saving, and running sets of forecasting jobs
- SCL source code for the *Forecast Command Builder*

SAS Software Products Needed

The Time Series Forecasting system is part of SAS/ETS software. To use it, you must have a license for SAS/ETS. To use the graphical display features of the system, you must also license SAS/GRAPH software.

Chapter 34
Getting Started

Chapter Contents

Chapter 34 Getting Started with Time Series Forecasting

This chapter outlines the forecasting process and introduces the major windows of the system through three example sessions.

The first example, beginning with the section "The Time Series Forecasting Window", shows how to use the system for fully automated forecasting of a set of time series. This example also introduces the system's features for viewing data and forecasts through tables and interactive graphs. It also shows how to save and restore forecasting work in SAS catalogs.

The second example, beginning with the section "Develop Models Window", introduces the features for developing the best forecasting models for individual time series. The chapter concludes with an example showing how to create dating variables for your data in the form expected by the system.

After working through the examples in this chapter, you should be able to

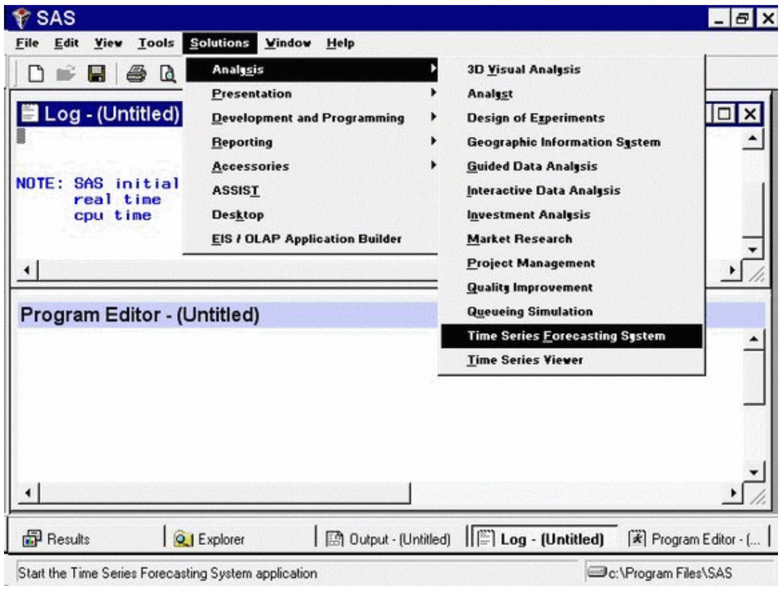
- select a data set of time series to work with and specify its periodicity and time ID variable
- use the automatic forecasting model selection feature to create forecasting models for the variables in a data set
- produce and save forecasts of variables in a data set
- examine your data and forecasts as tables of values and through interactive graphs
- save and restore your forecasting models using project files in a SAS catalog and edit project information
- use some of the model development features to fit and select forecasting models for individual time series variables.

This chapter introduces these topics and will help you get started using the system. Later chapters present these topics in greater detail and document more advanced features and options.

The Time Series Forecasting Window

There are several ways to get to the Time Series Forecasting System. If you prefer to use commands, invoke the system by entering `forecast` on the command line. You can optionally specify additional information on the command line; see Chapter 39, "Command Reference," for details.

If you are using the SAS Windowing Environment with pull-down menus, select the Solutions menu from the menu bar, select the Analysis item, and then select `Time Series Forecasting System`, as shown in Figure 34.1.



```
NOTE: SAS initial
      real time
      cpu time
```

Start the Time Series Forecasting System application

Figure 34.1. Time Series Forecasting System Menu Selection

You can invoke the Forecasting System from the SAS Explorer window by opening an existing forecasting project. By default these projects are stored in the FMSPROJ catalog in the SASUSER library. Select SASUSER in the Explorer to display its contents. Then select FMSPROJ. This catalog is created the first time you use the Forecasting System. If you have saved projects, they appear in the Explorer with the forecasting graph icon, as shown in Figure 34.2. Double-click one of the projects, or select it with the right mouse button and then select Open from the pop-up menu, as shown in the figure. This brings up the Forecasting System and opens the selected project.

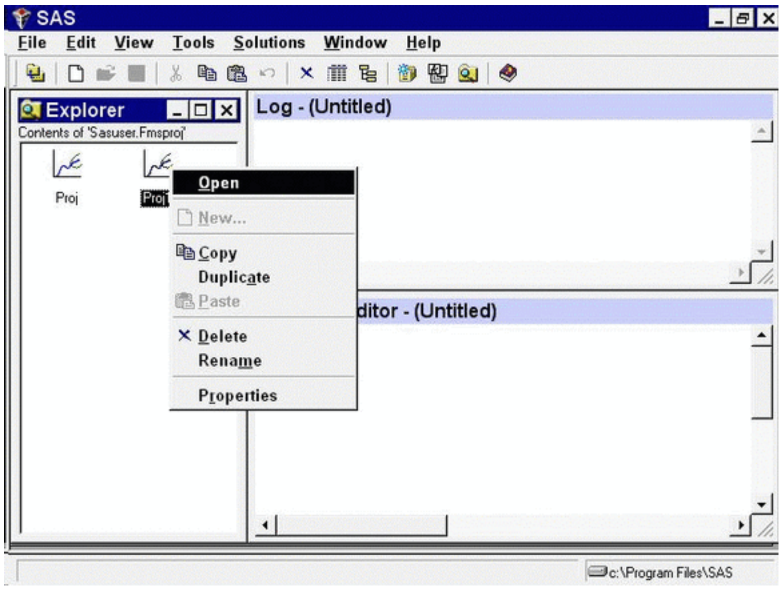


Figure 34.2. Opening a Project from the Explorer

To invoke the Forecasting System in the SAS Desktop environment, select the Solutions menu from the menu bar, select `Desktop`, and then open the Analysis folder. You can run the Time Series Forecasting System or the Time Series Viewer directly, or you can drag and drop. Figure 34.3 illustrates dragging a data set (known as a table in the Desktop environment) and dropping it on the Forecasting icon. In this example, the tables reside in a user-defined folder called *Time Series Data*.

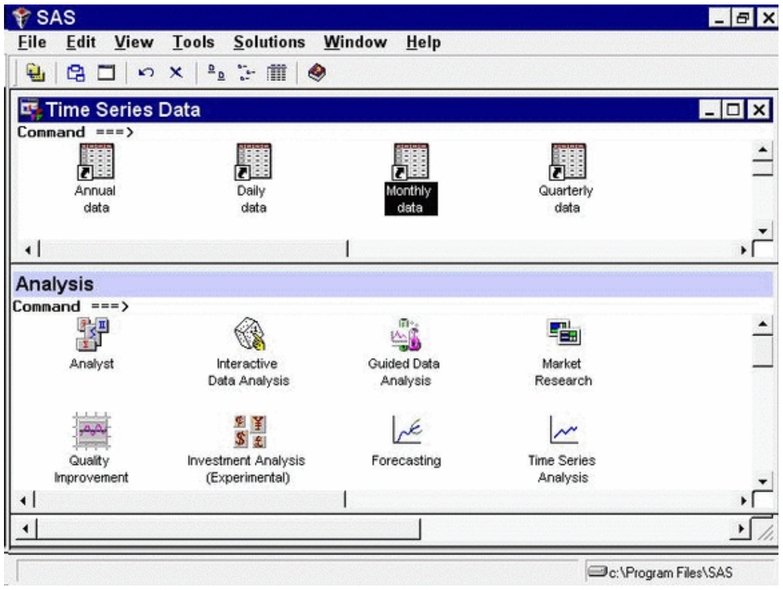
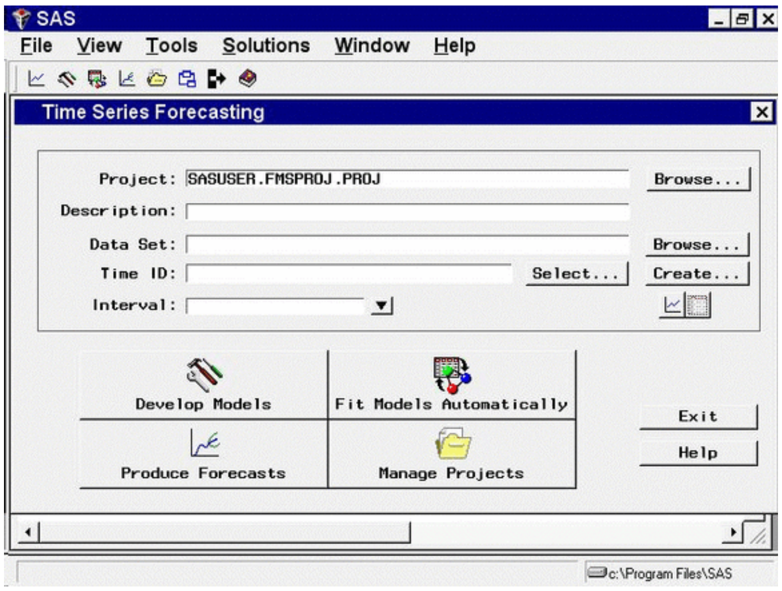


Figure 34.3. Drag and Drop Using the SAS Desktop

If you are using SAS/ASSIST software, select the Planning button and then select `Forecasting` from the pop-up menu.

Any of these methods takes you to the Time Series Forecasting window, as shown in Figure 34.4.



Project: SASUSER.FMSPROJ.PROJ
Description:
Data Set:
Time ID:
Interval:

Figure 34.4. Time Series Forecasting Window

At the top of the window is a data selection area for specifying a project file and the input data set containing historical data (the known past values) for the time series variables that you want to forecast. This area also contains buttons for bringing up viewers to explore your input data either graphically, one series at a time, or as a table, one data set at a time.

The Project and Description fields are used to specify a project file for saving and restoring forecasting models created by the system. Using project files is discussed later, and we will ignore these fields for now.

The lower part of the window contains six buttons:

`Develop Models`
: brings up the Develop Models window, which you use to develop and fit forecasting models interactively for individual time series.

`Fit Models Automatically`
: brings up the Automatic Model Fitting window, which you use to search automatically for the best forecasting model for multiple series in the input data set.

`Produce Forecasts`
: brings up the Produce Forecasts window, which you use to compute forecasts for all the variables in the input data set for which forecasting models have been fit.

`Manage Projects`
: brings up the Manage Forecasting Project window, which lists the time series for which you have fit forecasting models. You can drill down on a series to see the models that have been fit. You can delete series or models from the project, re-evaluate or refit models, and explore models and forecasts graphically or in tabular form.

`Exit`
: exits the Forecasting System.

`Help`
: displays information about the Forecasting System.

Outline of the Forecasting Process

The examples shown in the following sections illustrate the basic process you will use with the Forecasting System.

Specify the Input Data Set

Suppose you have a number of *time series*, variables recorded over time, for which you want to forecast future values. The past values of these time series are stored as variables in a SAS data set or data view. The observations of this data set correspond to regular time periods, such as days, weeks, or months. The first step in the forecasting process is to tell the system to use this data set by setting the Data Set field.

If your time series are not in a SAS data set, you must provide a way for the SAS System to access the data. You can use SAS features to read your data into a SAS data set; refer to *SAS Language Reference*. You can use a SAS/ACCESS product to establish a view of data in a database management system; refer to SAS/ACCESS documentation. You can use PROC SQL to create a SAS data view. You can use PROC DATASOURCE to read data from files supplied by supported data vendors; refer to Chapter 14, "The DATASOURCE Procedure," for more details.

Provide a Valid Time ID Variable

To use the Forecasting System, your data set must be dated: the data set must contain a *time ID variable* that gives the date of each observation. The time ID variable must represent the observation dates with *SAS date values* or with *SAS datetime values* (for hourly data or other frequencies less than a day), or you can use a simple time index.

When SAS date values are used, the ID variable contains dates within the time periods corresponding to the observations. For example, for monthly data, the values for the time ID variable may be the date of the first day of the month corresponding to each observation, or the time ID variable may contain the date of the last day in the month. (Any date within the period will serve as the time ID for the observation.)

If your data set already contains a valid time ID variable with SAS date or datetime values, the next step is to specify this time ID variable in the Time ID field. If the time ID variable is named DATE, the system fills in the Time ID field automatically.

If your data set does not contain a time ID, you must add a valid time ID variable before beginning the forecasting process. The Forecasting System provides features that make this easy to do. See Chapter 35, "Creating Time ID Variables," for details.

Select and Fit a Forecasting Model for each Series

If you are using the automated model selection feature, the system performs this step for you and chooses a forecasting model for each series automatically. All you need to do is select the Fit Models Automatically button and then select the variables to fit models for.

If you want more control over forecasting model selection, you can select the Develop Models button, select the series you want to forecast, and use the Develop Models window to specify a forecasting model. As part of this process, you may use the Time Series Viewer and Model Viewer graphical tools. Once you have selected a model for the first series, you can select a different series to work with and repeat the model development process until you have created forecasting models for all the series you want to forecast.

The system provides many features to help you choose the best forecasting model for each series. The features of the Develop Models window and graphical viewer tools are introduced in later sections.

Produce the Forecasts

Once a forecasting model has been fit for each series, select the Produce Forecasts button and use the Produce Forecasts window to compute forecast values and store them in a SAS data set.

Save Your Work

If you want only a single forecast, your task is now complete. But you may want to produce updated forecasts later, as more data becomes available. In this case, you want to save the forecasting models you have created, so that you will not need to repeat the model selection and fitting process.

To save your work, fill in the Project field with the name of a SAS catalog member in which the system will store the model information when you exit the system. Later, you will select the same catalog member name when you first enter the Forecasting System, and the model information will be reloaded.

Note that any number of people can work with the same project file. If you are working on a forecasting project as part of a team, you should take care to avoid conflicting updates to the project file by different team members.

Summary

This is the basic outline of how the Forecasting System works. The system offers many other features and options that you may need to use (for example, the time range of the data used to fit models and how far into the future to forecast). These options will become apparent as you work with the Forecasting System.

As an introductory example, the following sections use the Automatic Model Fitting and Produce Forecasts windows to perform automated forecasting of the series in an example data set.

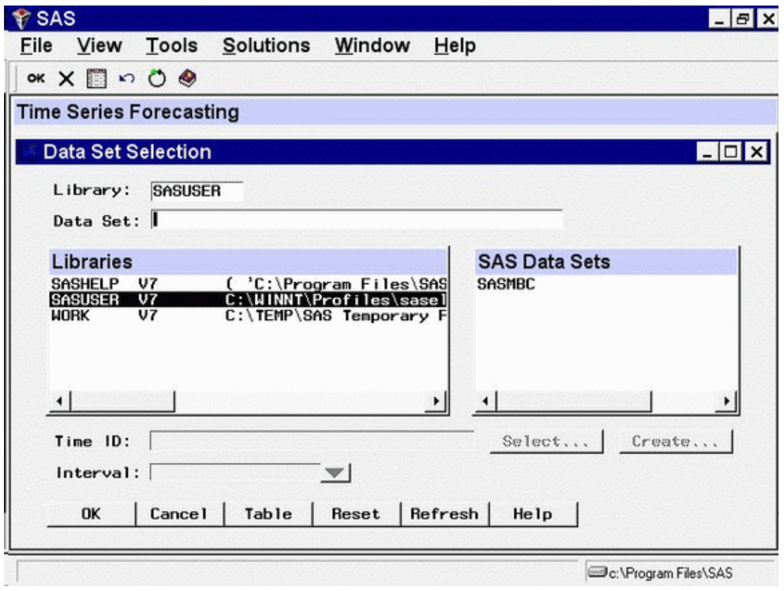
The Input Data Set

As the first step, you must specify the input data set.

The Data Set field in the Time Series Forecasting window gives the name of the input data set containing the time series to forecast. Initially, this field is blank. You can specify the input data set by typing the data set name in this field. Alternatively, you can select the Browse button at the right of the Data Set field to select the data set from a list, as shown in the following section.

The Data Set Selection Window

Select the Browse button to the right of the Data Set field. This brings up the Data Set Selection window, as shown in Figure 34.5.



Library: SASUSER
Data Set:

Libraries

```
SASHELP  V7     ( 'C:\Program Files\SAS
SASUSER  V7     C:\WINNT\Profiles\sasel
WORK     V7     C:\TEMP\SAS Temporary F
```

SAS Data Sets

```
SASMBC
```

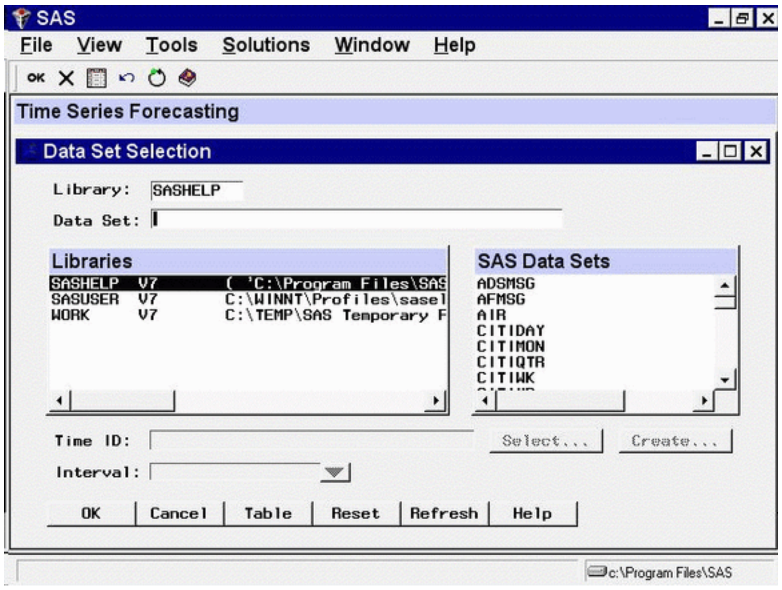
Time ID:
Interval:

c:\Program Files\SAS

Figure 34.5. Data Set Selection Window

The `Libraries` list shows the SAS librefs that are currently allocated in your SAS session. Initially, the SASUSER library is selected, and the `SAS Data Sets` list shows the data sets available in your SASUSER library.

In the `Libraries` list, select the row that starts with SASHELP. The Data Set Selection window now lists the data sets in the SASHELP library, as shown in Figure 34.6.

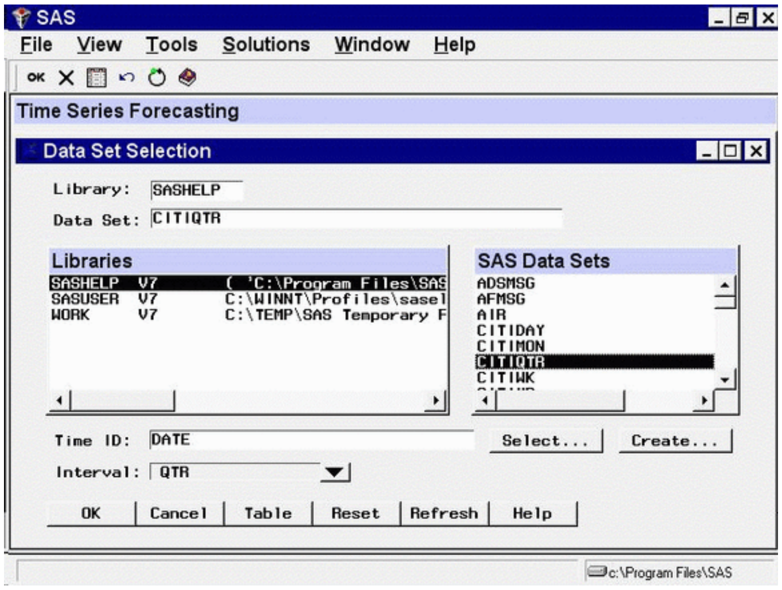


```
Library:  SASHELP
Data Set:

Libraries
SASHELP   V7      ( 'C:\Program Files\SAS
SASUSER   V7      C:\WINNT\Profiles\sasel
WORK      V7      C:\TEMP\SAS Temporary F

SAS Data Sets
ADSMSG
AFMSG
AIR
CITIDAY
CITIMON
CITIQTR
CITIWK

Time ID:
Interval:

c:\Program Files\SAS
```

Figure 34.6. SASHELP Library

Use the vertical scroll bar on the SAS Data Sets list to scroll down the list until the data set CITIQTR appears. Then select the CITIQTR row. This selects the data set SASHELP.CITIQTR as the input data set.

Figure 34.7 shows the Data Set Selection window after selection of CITIQTR from the SAS Data Sets list.

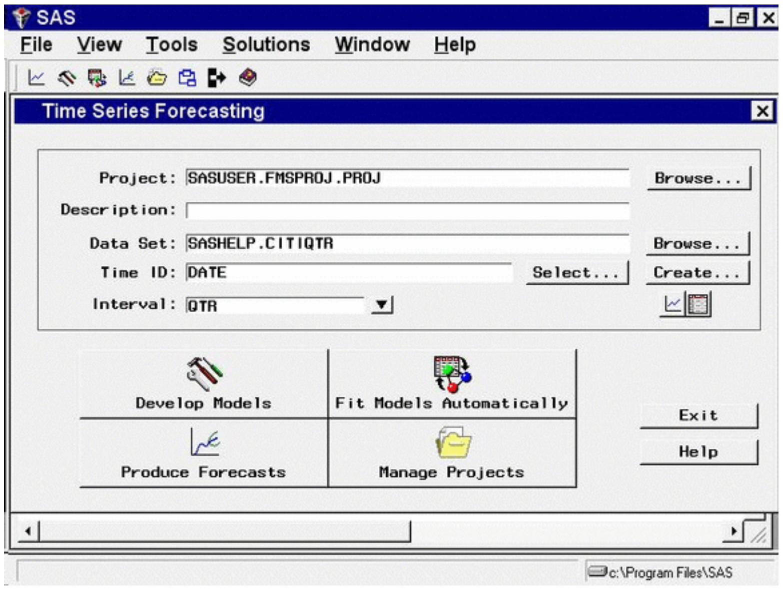


```
Library:  SASHELP
Data Set: CITIQTR

Libraries
SASHELP   V7      ( 'C:\Program Files\SAS
SASUSER   V7      C:\WINNT\Profiles\sasel
WORK      V7      C:\TEMP\SAS Temporary F

SAS Data Sets
ADSMSG
AFMSG
AIR
CITIDAY
CITIMON
CITIQTR
CITIWK

Time ID:  DATE
Interval: QTR

c:\Program Files\SAS
```

Figure 34.7. CITIQTR Data Set Selected

Note that the Time ID field is now set to DATE and the Interval field is set to QTR. These fields are explained in the following section.

Now select the OK button to complete selection of the CITIQTR data set. This closes the Data Set Selection window and returns to the Time Series Forecasting window, as shown in Figure 34.8.



Project: SASUSER.FMSPROJ.PROJ
Description:
Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR

Figure 34.8. Time Series Forecasting Window

Time Series Data Sets, ID variables, and Time Intervals

Before you continue with the example, it is worthwhile to consider how the system determined the values for the Time ID and Interval fields in the Data Set Selection window.

The Forecasting System requires that the input data set contain time series observations, with one observation for each time period. The observations must be sorted in increasing time order, and there must be no gaps in the sequence of observations. The time period of each observation must be identified by an ID variable, which is shown in the Time ID field.

If the data set contains a variable named DATE, TIME, or DATETIME, the system assumes that that variable is the SAS date or datetime valued ID variable, and the Time ID field is filled in automatically. The time ID variable for the SASHELP.CITIQTR data set is named DATE, and therefore the system set the Time ID field to DATE.

If the time ID variable for a data set is not named DATE, TIME, or DATETIME, you must specify the time ID variable name. You can specify the time ID variable either by typing the ID variable name in the Time ID field or by clicking the Select button.

If your data set does not contain a time ID variable with SAS date values, you can add a time ID variable using one of the windows described in Chapter 35, “Creating Time ID Variables.”

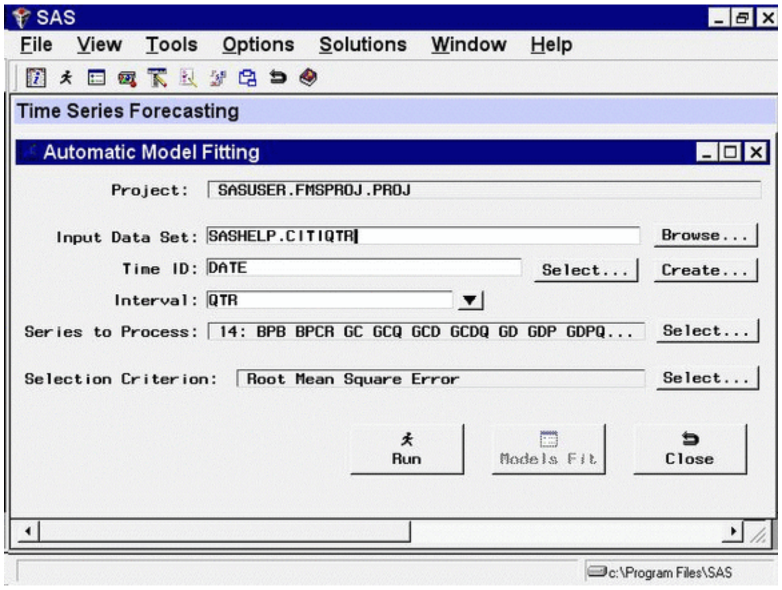
Once the time ID variable is known, the Forecasting System examines the ID values to determine the time interval between observations. The data set SASHELP.CITIQTR contains quarterly observations. Therefore, the system determined that the data have a quarterly interval, and set the Interval field to QTR.

If the system cannot determine the data frequency from the values of the time ID variable, you must specify the time interval between observations. You can specify the time interval using the `Interval` combo box. In addition to the interval names provided in the pop-up list, you can type in more complex interval names to specify an interval that is a multiple of other intervals or that has date values in the middle of the interval (such as monthly data with time ID values falling on the 10th day of the month).

See Chapter 2, "Working with Time Series Data," and Chapter 3, "Date Intervals, Formats, and Functions," for more information on time intervals, SAS date values, and ID variables for time series data sets.

Automatic Model Fitting Window

Before you can produce forecasts, you must fit forecasting models to the time series. Select the Fit Models Automatically button. This brings up the Automatic Model Fitting window, as shown in Figure 34.9.



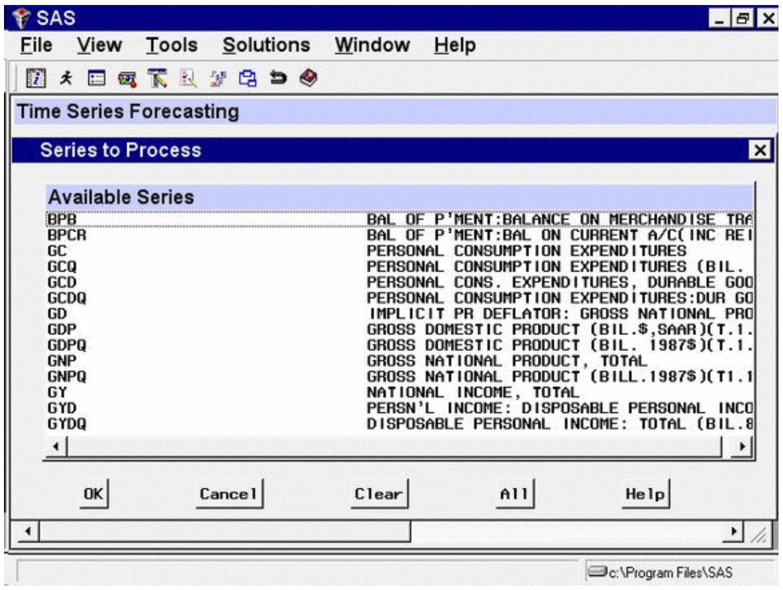
Project: SASUSER.FMSPROJ.PROJ
Input Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR
Series to Process: 14: BPB BPCR GC GCQ GCD GCDQ GD GDP GDPQ...
Selection Criterion: Root Mean Square Error

Figure 34.9. Automatic Model Fitting Window

The first part of the Automatic Model Fitting window confirms the project filename and the input data set name.

The Series to Process field shows the number and lists the names of the variables in the input data set to which the Automatic Model Fitting process will be applied. By default, all numeric variables (except the time ID variable) are processed. However, you can specify that models be generated for only a select subset of these variables.

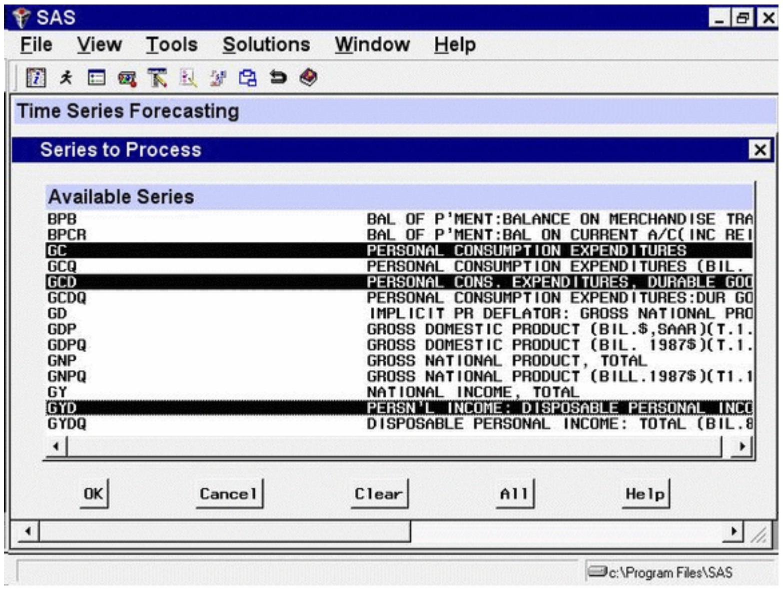
Click the Select button to the right of the Series to Process field. This brings up the Series to Process window, as shown in Figure 34.10.



```
BPB     BAL OF P'MENT:BALANCE ON MERCHANDISE TRA
BPCR    BAL OF P'MENT:BAL ON CURRENT A/C( INC REI
GC      PERSONAL CONSUMPTION EXPENDITURES
GCQ     PERSONAL CONSUMPTION EXPENDITURES (BIL.
GCD     PERSONAL CONS. EXPENDITURES, DURABLE GOO
GCDQ    PERSONAL CONSUMPTION EXPENDITURES:DUR GO
GD      IMPLICIT PR DEFLATOR: GROSS NATIONAL PRO
GDP     GROSS DOMESTIC PRODUCT (BIL.$,SAAR)(T.1.
GDPQ    GROSS DOMESTIC PRODUCT (BIL. 1987$)(T.1.
GNP     GROSS NATIONAL PRODUCT, TOTAL
GNPQ    GROSS NATIONAL PRODUCT (BILL.1987$)(T1.1
GY      NATIONAL INCOME, TOTAL
GYD     PERSN'L INCOME: DISPOSABLE PERSONAL INCO
GYDQ    DISPOSABLE PERSONAL INCOME: TOTAL (BIL.8
```

Figure 34.10. Series to Process Window

Use the mouse and the CTRL key to select the personal consumption expenditures series (GC), the personal consumption expenditures for durable goods series (GCD), and the disposable personal income series (GYD), as shown in Figure 34.11. (Remember to hold down the CTRL key as you make the selections; otherwise, selecting a second series will deselect the first.)



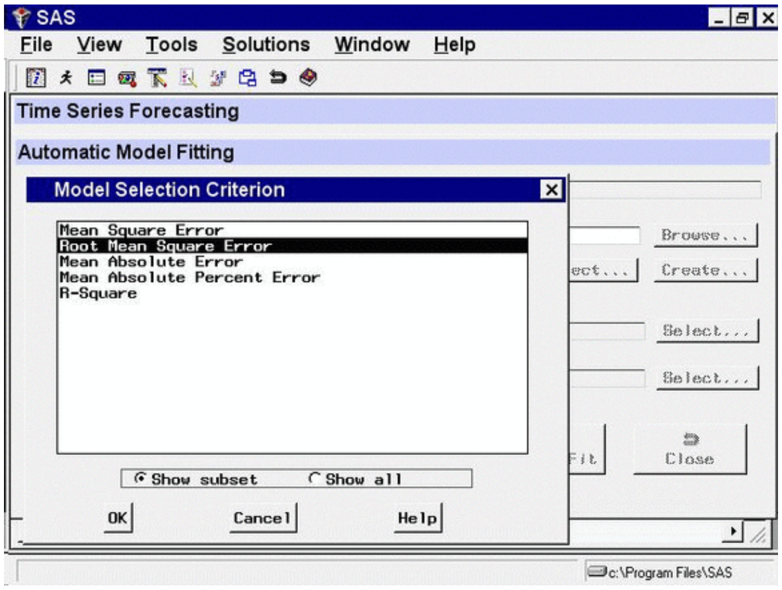
```
BPB     BAL OF P'MENT:BALANCE ON MERCHANDISE TRA
BPCR    BAL OF P'MENT:BAL ON CURRENT A/C( INC REI
GC      PERSONAL CONSUMPTION EXPENDITURES
GCQ     PERSONAL CONSUMPTION EXPENDITURES (BIL.
GCD     PERSONAL CONS. EXPENDITURES, DURABLE GOO
GCDQ    PERSONAL CONSUMPTION EXPENDITURES:DUR GO
GD      IMPLICIT PR DEFLATOR: GROSS NATIONAL PRO
GDP     GROSS DOMESTIC PRODUCT (BIL.$,SAAR)(T.1.
GDPQ    GROSS DOMESTIC PRODUCT (BIL. 1987$)(T.1.
GNP     GROSS NATIONAL PRODUCT, TOTAL
GNPQ    GROSS NATIONAL PRODUCT (BILL.1987$)(T1.1
GY      NATIONAL INCOME, TOTAL
GYD     PERSN'L INCOME: DISPOSABLE PERSONAL INCO
GYDQ    DISPOSABLE PERSONAL INCOME: TOTAL (BIL.8
```

Figure 34.11. Selecting Series for Automatic Model Fitting

Now select the OK button. This returns you to the Automatic Model Fitting window. The Series to Process field now shows the selected variables.

The Selection Criterion field shows the goodness-of-fit measure that the Forecasting System will use to select the best fitting model for each series. By default, the selection criterion is the root mean square error. To illustrate how you can control the selection criterion, this example will use the mean absolute percent error to select the best fitting models.

Click the Select button to the right of the Selection Criterion field. This brings up a list of statistics of fit, as shown in Figure 34.12.

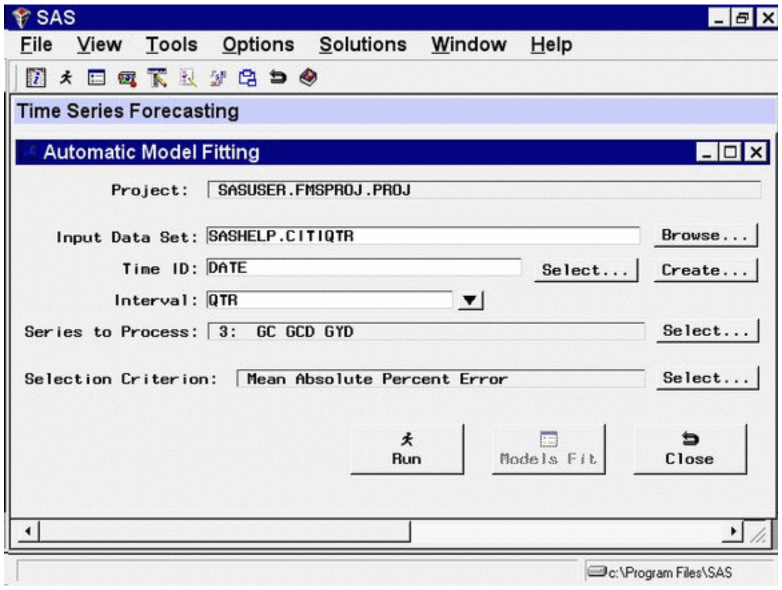


Mean Square Error
Root Mean Square Error
Mean Absolute Error
Mean Absolute Percent Error
R-Square

[✓ Show subset] Show all

Figure 34.12. Choosing the Model Selection Criterion

Select *Mean Absolute Percent Error* and then select the OK button. The Automatic Model Fitting window now appears as shown in Figure 34.13.

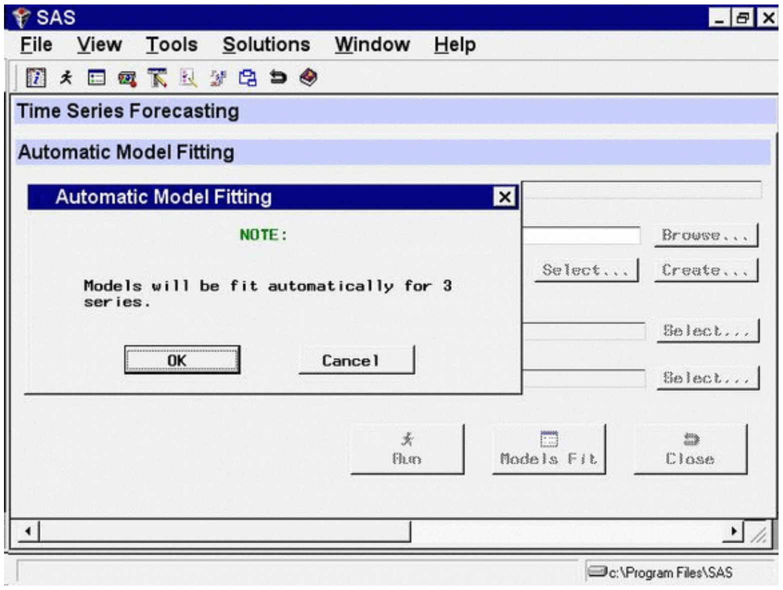


Project: SASUSER.FMSPROJ.PROJ
Input Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR
Series to Process: 3: GC GCD GYD
Selection Criterion: Mean Absolute Percent Error

Figure 34.13. Automatic Model Fitting Window

Now that all the options are set appropriately, select the Run button.

The Forecasting System now displays a notice, shown in Figure 34.14, confirming that models will be fit for 3 series using the automatic forecasting model search feature. This prompt is displayed because it is possible to fit models for a large number of series at once, which may take a lot of time, and so the system gives you a chance to cancel if you accidentally ask to fit models for more series than you intended. Select the OK button.

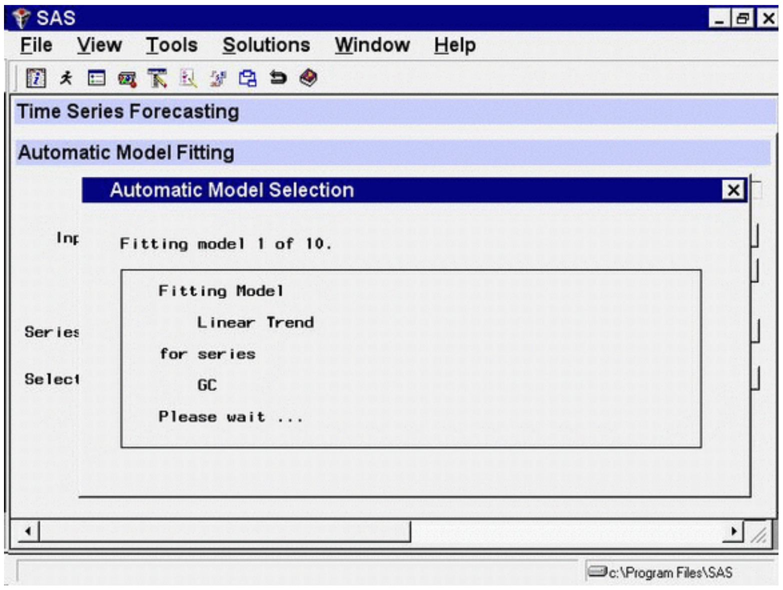


NOTE:

Models will be fit automatically for 3 series.

Figure 34.14. Automatic Model Fitting Note

The system now fits several forecasting models to each of the three series you selected. While the models are being fit, the Forecasting System displays notices indicating what it is doing so that you can observe its progress, as shown in Figure 34.15.



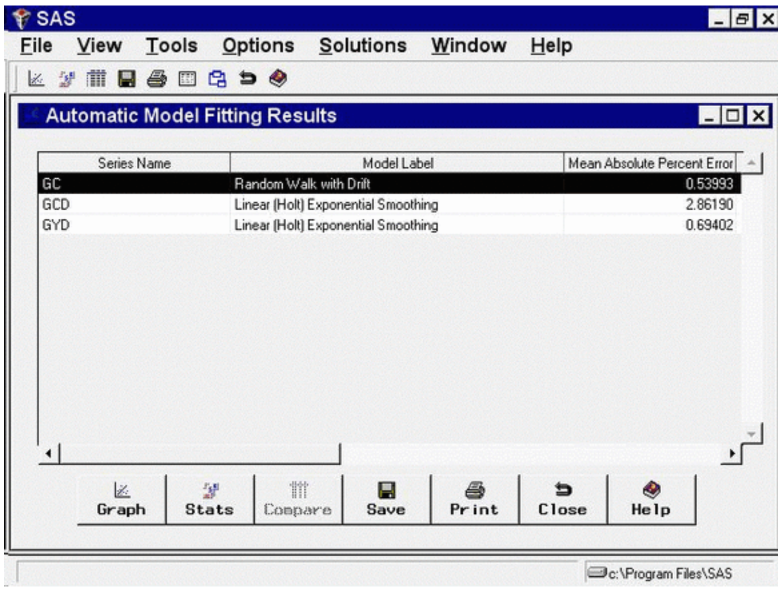
Fitting model 1 of 10.

Fitting Model
Linear Trend
for series
GC
Please wait ...

Figure 34.15. "Working" Notice

For each series, the system saves the model that produces the smallest mean absolute percent error. You can have the system save all the models fit by selecting `Automatic Fit` from the Options pull-down menu.

After the Automatic Model Fitting process has completed, the results are displayed in the Automatic Model Fitting Results window, as shown in Figure 34.16.


SAS
File View Tools Options Solutions Window Help
Automatic Model Fitting Results
Graph
Stats
Compare
Save
Print
Close
Help
c:\Program Files\SAS

Series Name	Model Label	Mean Absolute Percent Error
GC	Random Walk with Drift	0.53993
GCD	Linear (Holt) Exponential Smoothing	2.86190
GYD	Linear (Holt) Exponential Smoothing	0.69402

Figure 34.16. Automatic Model Fitting Results

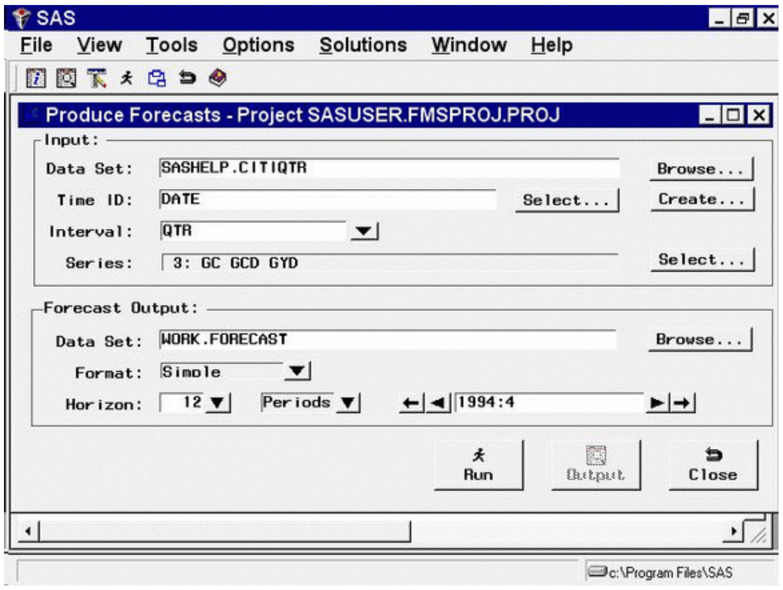
This resizable window shows the list of series names and descriptive labels for the forecasting models chosen for them, as well as the values of the model selection criterion and other statistics of fit. Select the Close button.

This returns you to the Automatic Model Fitting window. You can now fit models for other series in this data set or change to a different data set and fit models for series in the new data set.

Select the Close button to return to the Time Series Forecasting window.

Produce Forecasts Window

Now that you have forecasting models for these three series, you are ready to produce forecasts. Select the Produce Forecasts button. This brings up the Produce Forecasts window, as shown in Figure 34.17.



Produce Forecasts - Project SASUSER.FMSPROJ.PROJ

Input:
Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR
Series: 3: GC GCD GYD

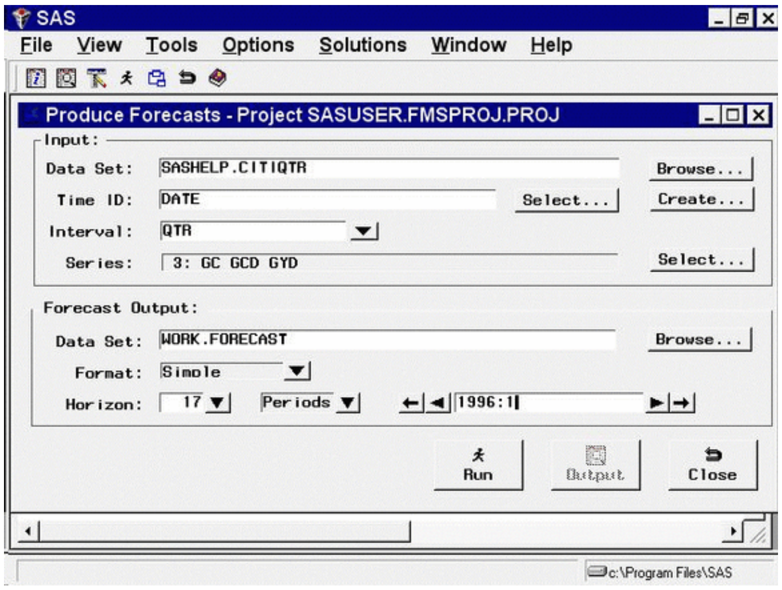
Forecast Output:
Data Set: WORK.FORECAST
Format: Simple
Horizon: 12 Periods 1994:4

Figure 34.17. Produce Forecasts Window

The Produce Forecasts window shows the input data set information and indicates the variables in the input data set for which forecasting models exist. Forecasts will be produced for these series. If you want to produce forecasts for only some of these series, use the control arrow at the right of the Series field to select the series to forecast. The Data Set field in the `Forecast Output` box contains the name of the SAS data set in which the system will store the forecasts. The default output data set is WORK.FORECAST.

You can set the forecast horizon using the controls on the line labeled `Horizon`. The default horizon is 12 periods. You can change it using number of periods, number of years, or the date of the last forecast period. Position the cursor in the date field and change the forecast ending date to 1 January 1996 by typing `jan1996` and pressing the ENTER key.

The window now appears as shown in Figure 34.18.



Produce Forecasts - Project SASUSER.FMSPROJ.PROJ

Input:
Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR
Series: 3: GC GCD GYD

Forecast Output:
Data Set: WORK.FORECAST
Format: Simple
Horizon: 17 Periods 1996:1

Figure 34.18. Produce Forecasts Window

Now select the Run button to produce the forecasts. The system indicates that the forecasts have been stored in the output data set. Select `OK` to dismiss the notice.

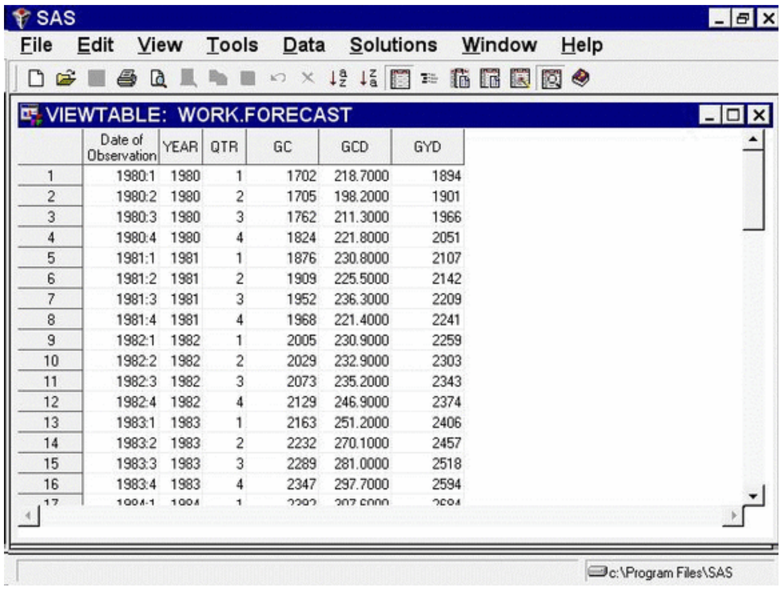
The Forecast Data Set

The Forecasting System can save the forecasts to a SAS data set in three different formats. Depending on your needs, you may find one of these output formats more convenient. The output data set format is controlled by the `Format` combo box. You can select the following output formats. The simple format is the default.

`Simple`	The data set contains time ID variables and the forecast variables, and it contains one observation per time period. Observations for earlier time periods contain actual values copied from the input data set; later observations contain the forecasts.
`Interleaved`	The data set contains time ID variables, the variable TYPE, and the forecast variables. There are several observations per time period, with the meaning of each observation identified by the TYPE variable.
`Concatenated`	The data set contains the variable SERIES, time ID variables, and the variables ACTUAL, PREDICT, ERROR, UPPER, LOWER, and STD. There is one observation per time period per forecast series. The variable SERIES contains the name of the forecast series, and the data set is sorted by SERIES and DATE.

Simple Format Forecast Data Set

To see the simple format forecast data set that the system created, select the Output button. This brings up a Viewtable window to display the data set, as shown in Figure 34.19.



	Date of Observation	YEAR	QTR	GC	GCD	GYD
1	1980:1	1980	1	1702	218.7000	1894
2	1980:2	1980	2	1705	198.2000	1901
3	1980:3	1980	3	1762	211.3000	1966
4	1980:4	1980	4	1824	221.8000	2051
5	1981:1	1981	1	1876	230.8000	2107
6	1981:2	1981	2	1909	225.5000	2142
7	1981:3	1981	3	1952	236.3000	2209
8	1981:4	1981	4	1968	221.4000	2241
9	1982:1	1982	1	2005	230.9000	2259
10	1982:2	1982	2	2029	232.9000	2303
11	1982:3	1982	3	2073	235.2000	2343
12	1982:4	1982	4	2129	246.9000	2374
13	1983:1	1983	1	2163	251.2000	2406
14	1983:2	1983	2	2232	270.1000	2457
15	1983:3	1983	3	2289	281.0000	2518
16	1983:4	1983	4	2347	297.7000	2594
17	1984:1	1984	1	[illegible]	[illegible]	[illegible]

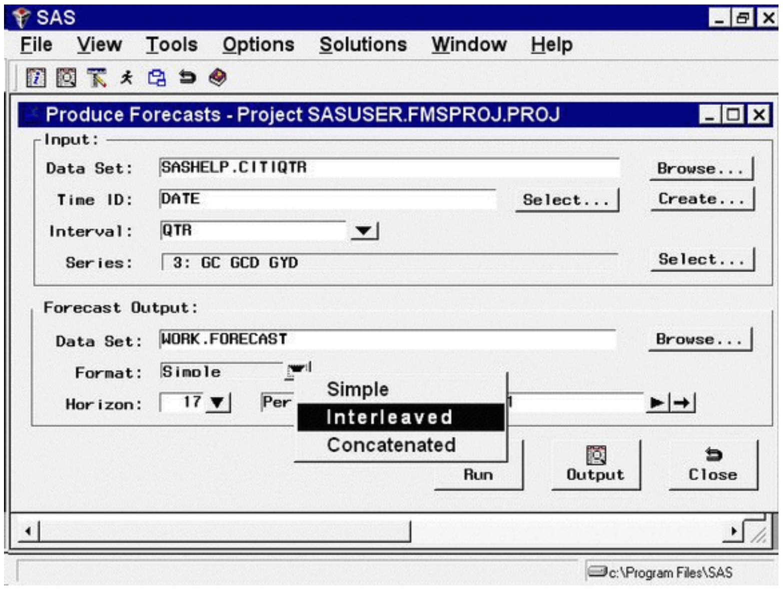
Figure 34.19. Forecast Data Set – Simple Format

Figure 34.19 shows the default simple format. This form of the forecast data set contains time ID variables and the variables that you forecast. The forecast variables contain actual values or predicted values, depending on whether the date of the observation is within the range of data supplied in the input data set.

Select `File` and `Close` to close the Viewtable window.

Interleaved Format Forecast Data Set

From the Produce Forecasts window, use the combo box to select the `Interleaved` format, as shown in Figure 34.20.



Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR
Series: 3: GC GCD GYD

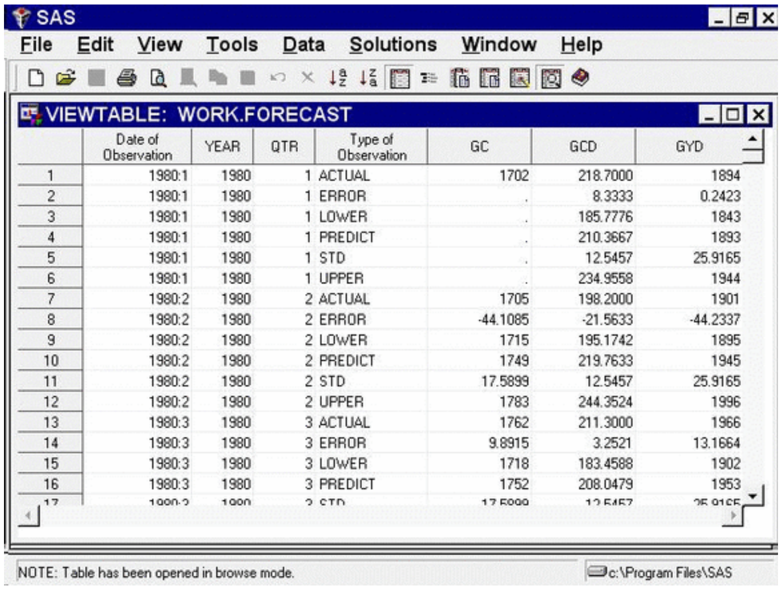
Data Set: WORK.FORECAST
Format: Simple
Horizon: 17 Per

Figure 34.20. Forecast Data Set Options

Now select the Run button again. The system presents a warning notice reminding you that the data set WORK.FORECAST already exists and asking if you want to replace it. Select `Replace`.

The forecasts are stored in the data set WORK.FORECAST again, this time in the *Interleaved* format. Dismiss the notice that the forecast was stored.

Now select the Output button again. This brings up a Viewtable window to display the data set, as shown in Figure 34.21.



VIEWTABLE: WORK.FORECAST

	Date of Observation	YEAR	QTR	Type of Observation	GC	GCD	GYD
1	1980:1	1980	1	ACTUAL	1702	218.7000	1894
2	1980:1	1980	1	ERROR	.	8.3333	0.2423
3	1980:1	1980	1	LOWER	.	185.7776	1843
4	1980:1	1980	1	PREDICT	.	210.3667	1893
5	1980:1	1980	1	STD	.	12.5457	25.9165
6	1980:1	1980	1	UPPER	.	234.9558	1944
7	1980:2	1980	2	ACTUAL	1705	198.2000	1901
8	1980:2	1980	2	ERROR	-44.1085	-21.5633	-44.2337
9	1980:2	1980	2	LOWER	1715	195.1742	1895
10	1980:2	1980	2	PREDICT	1749	219.7633	1945
11	1980:2	1980	2	STD	17.5899	12.5457	25.9165
12	1980:2	1980	2	UPPER	1783	244.3524	1996
13	1980:3	1980	3	ACTUAL	1762	211.3000	1966
14	1980:3	1980	3	ERROR	9.8915	3.2521	13.1664
15	1980:3	1980	3	LOWER	1718	183.4588	1902
16	1980:3	1980	3	PREDICT	1752	208.0479	1953
17	1980:3	1980	3	STD	17.5899	12.5457	25.9165

NOTE: Table has been opened in browse mode.

c:\Program Files\SAS

Figure 34.21. Forecast Data Set – Interleaved Format

In the interleaved format, there are several output observations for each input observation, identified by the TYPE variable. The values of the forecast variables for observations with different TYPE values are as follows.

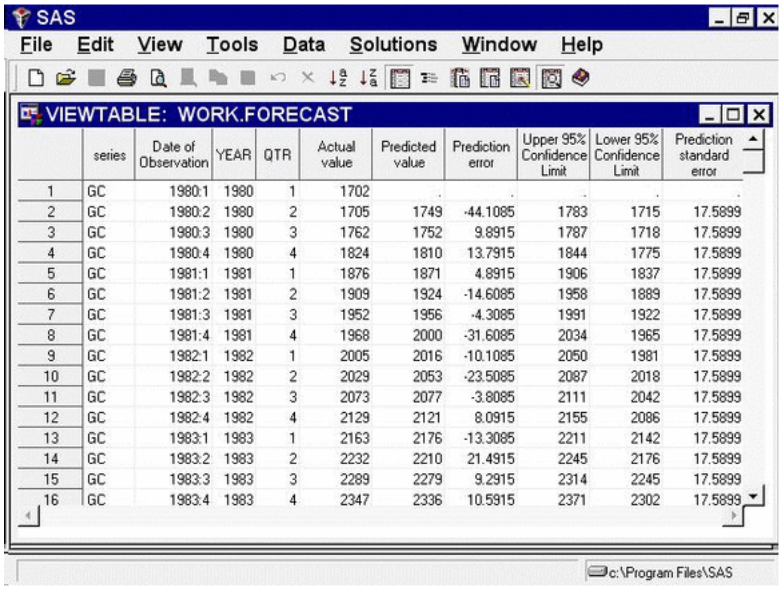
`ACTUAL`	actual values copied from the input data set.
`ERROR`	the difference between the actual and predicted values.
`LOWER`	the lower confidence limits.
`PREDICT`	the predicted values from the forecasting model. These are within-sample, one-step-ahead predictions for observations within the historical period, or multistep predictions for observations within the forecast period.
`STD`	the estimated standard deviations of the prediction errors.
`UPPER`	the upper confidence limits.

Select `File` and `Close` to close the Viewtable window.

Concatenated Format Forecast Data Set

Use the combo box to select the `Concatenated` format. Re-create the forecast data set again, and then select the Output button.

The Viewtable window showing the concatenated format of the forecast data set appears, as shown in Figure 34.22.



	series	Date of Observation	YEAR	QTR	Actual value	Predicted value	Prediction error	Upper 95% Confidence Limit	Lower 95% Confidence Limit	Prediction standard error
1	GC	1980:1	1980	1	1702	.	.	.	.	.
2	GC	1980:2	1980	2	1705	1749	-44.1085	1783	1715	17.5899
3	GC	1980:3	1980	3	1762	1752	9.8915	1787	1718	17.5899
4	GC	1980:4	1980	4	1824	1810	13.7915	1844	1775	17.5899
5	GC	1981:1	1981	1	1876	1871	4.8915	1906	1837	17.5899
6	GC	1981:2	1981	2	1909	1924	-14.6085	1958	1889	17.5899
7	GC	1981:3	1981	3	1952	1956	-4.3085	1991	1922	17.5899
8	GC	1981:4	1981	4	1968	2000	-31.6085	2034	1965	17.5899
9	GC	1982:1	1982	1	2005	2016	-10.1085	2050	1981	17.5899
10	GC	1982:2	1982	2	2029	2053	-23.5085	2087	2018	17.5899
11	GC	1982:3	1982	3	2073	2077	-3.8085	2111	2042	17.5899
12	GC	1982:4	1982	4	2129	2121	8.0915	2155	2086	17.5899
13	GC	1983:1	1983	1	2163	2176	-13.3085	2211	2142	17.5899
14	GC	1983:2	1983	2	2232	2210	21.4915	2245	2176	17.5899
15	GC	1983:3	1983	3	2289	2279	9.2915	2314	2245	17.5899
16	GC	1983:4	1983	4	2347	2336	10.5915	2371	2302	17.5899

c:\Program Files\SAS

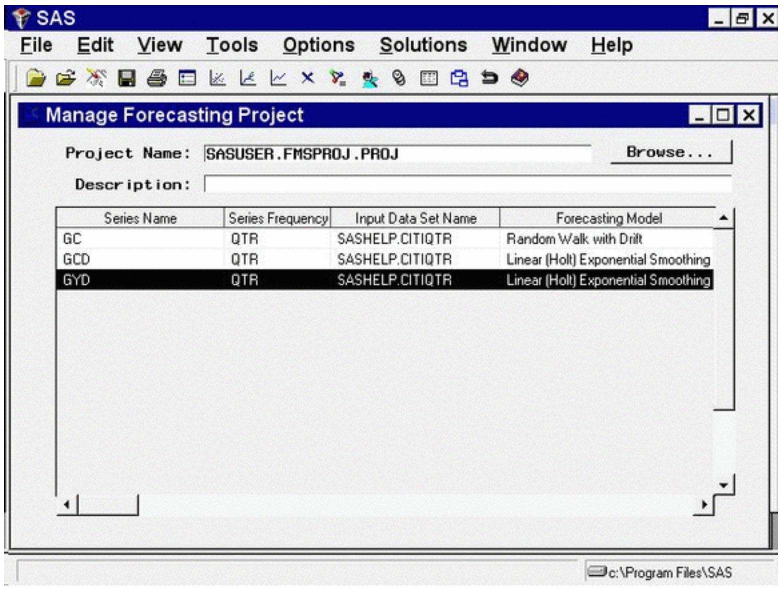
Figure 34.22. Forecast Data Set – Concatenated Format

This completes the example of how to use the Produce Forecasts window. Select `File` and `Close` to close the Viewtable window. Select the Close button to return to the Time Series Forecasting window.

Forecasting Projects

The system collects all the forecasting models you create, together with the options you set, into a package called a *forecasting project*. You can save this information in a SAS catalog entry and restore your work in later forecasting sessions. You can store any number of forecasting projects under different catalog entry names.

To see how this works, select the Manage Projects button. This brings up the Manage Forecasting Project window, as shown in Figure 34.23.


SAS
File Edit View Tools Options Solutions Window Help
Manage Forecasting Project
Browse...
c:\Program Files\SAS

Project Name: SASUSER.FMSPROJ.PROJ

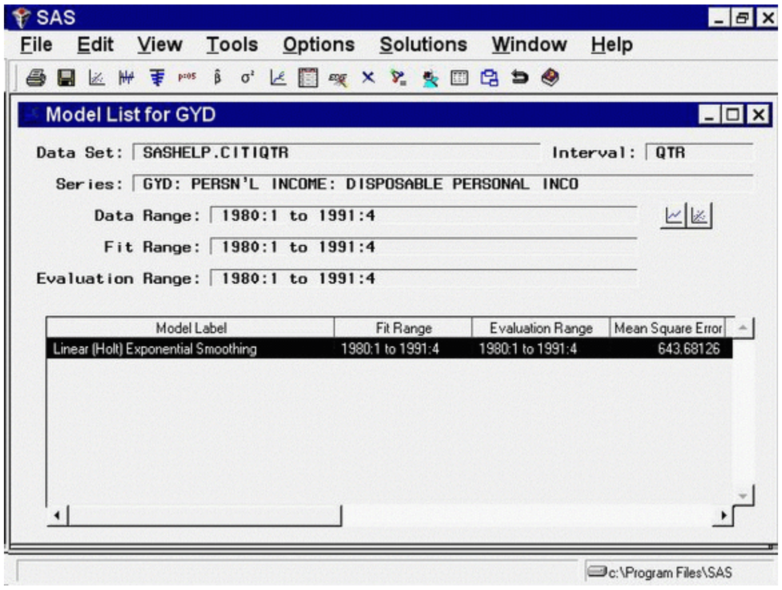
Description:

Series Name	Series Frequency	Input Data Set Name	Forecasting Model
GC	QTR	SASHELP.CITIQTR	Random Walk with Drift
GCD	QTR	SASHELP.CITIQTR	Linear (Holt) Exponential Smoothing
GYD	QTR	SASHELP.CITIQTR	Linear (Holt) Exponential Smoothing

Figure 34.23. Manage Forecasting Project Window

The table in this window lists the series for which forecasting models have been fit, and it shows for each series the forecasting model used to produce the forecasts. This window provides several features that allow you to manage the information in your forecasting project.

You can select a row of the table to drill down to the list of models fit to the series. Select the GYD row of the table, either by double-clicking with the mouse or by clicking once to highlight the table row and then selecting `List Models` from the toolbar or from the Tools pull-down menu. This brings up the Model List window for this series, as shown in Figure 34.24.



Data Set: SASHELP.CITIQTR Interval: QTR

Series: GYD: PERSN'L INCOME: DISPOSABLE PERSONAL INCO

Data Range: 1980:1 to 1991:4

Fit Range: 1980:1 to 1991:4

Evaluation Range: 1980:1 to 1991:4

Model Label	Fit Range	Evaluation Range	Mean Square Error
Linear (Holt) Exponential Smoothing	1980:1 to 1991:4	1980:1 to 1991:4	643.68126

Figure 34.24. Model List Window

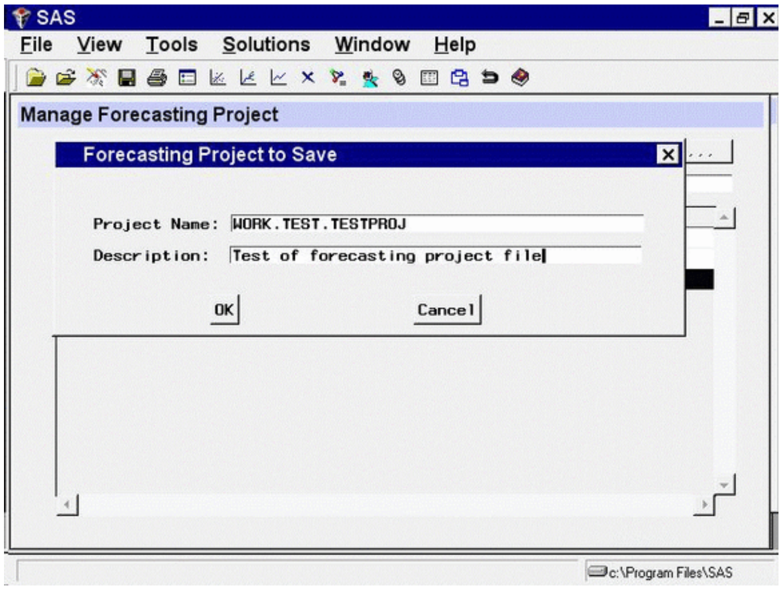
Because the Automatic Model Fitting Process process kept only the best fitting model, only one model appears in the model list. You can fit and retain any number of models for each series, and all the models fit and kept will appear in the series' model list.

Select `Close` from the toolbar or from the File pull-down menu to return to the Manage Forecasting Project window.

Saving and Restoring Project Information

To illustrate how you can save your work between sessions, in this section you will exit and then re-enter the Forecasting System.

From the Manage Forecasting Project window, select `File` and `Save as`. This brings up the Forecasting Project to Save window. In the Project Name field type the name WORK.TEST.TESTPROJ. In the Description field, type "Test of forecasting project file". The window should now appear as shown in Figure 34.25.


SAS
File View Tools Solutions Window Help
Manage Forecasting Project
Forecasting Project to Save
OK
Cancel
c:\Program Files\SAS

Project Name: WORK.TEST.TESTPROJ
Description: Test of forecasting project file

Figure 34.25. Project to Save Name and Description

Select the OK button. This returns you to the Project Management window, and displays a message indicating that the project was saved.

Select `Close` from the tool-bar or from the File pull-down menu to return to the Time Series Forecasting window. Now select the Exit button. The system asks if you are sure you want to exit the system; select `Yes`. The forecasting application now terminates.

Bring up the forecasting application again. A new project name is displayed by default.

Now restore the forecasting project you saved previously. Select the Browse button to the right of the Project field. This brings up the Forecasting Project File Selection window, as shown in Figure 34.26.

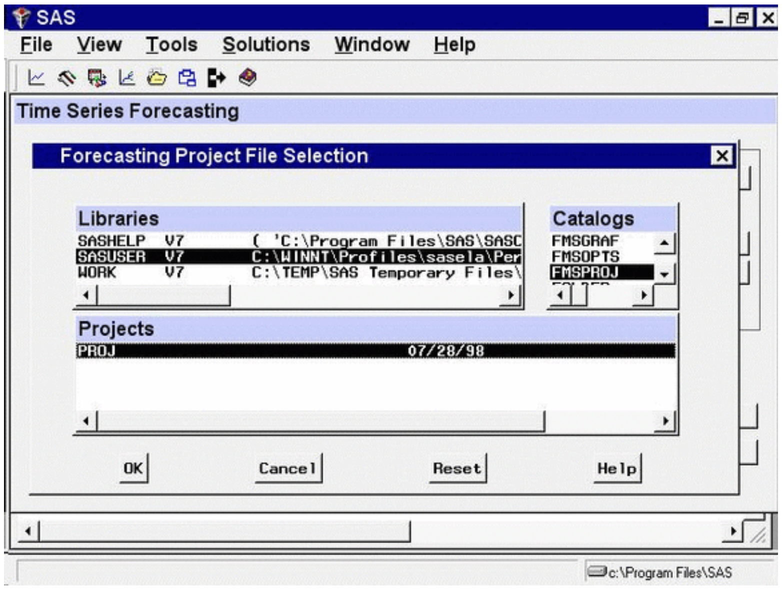

SAS
File View Tools Solutions Window Help
Time Series Forecasting
Forecasting Project File Selection
Libraries
SASHELP V7 ('C:\Program Files\SAS\SASC
SASUSER V7 C:\WINNT\Profiles\sasela\Per
WORK V7 C:\TEMP\SAS Temporary Files\
Catalogs
FMSGRAF
FMSOPTS
FMSPROJ
Projects
PROJ 07/28/98
OK
Cancel
Reset
Help
c:\Program Files\SAS

Figure 34.26. Forecasting Project File Selection Window

Select the WORK library from the Libraries list. The Catalogs list now shows all the SAS catalogs in the WORK library.

Select the TEST catalog. The Projects list now shows the list of forecasting projects in the catalog TEST. So far, you have created only one project file, TESTPROJ, so TESTPROJ is the only entry in the Projects list, as shown in Figure 34.27.

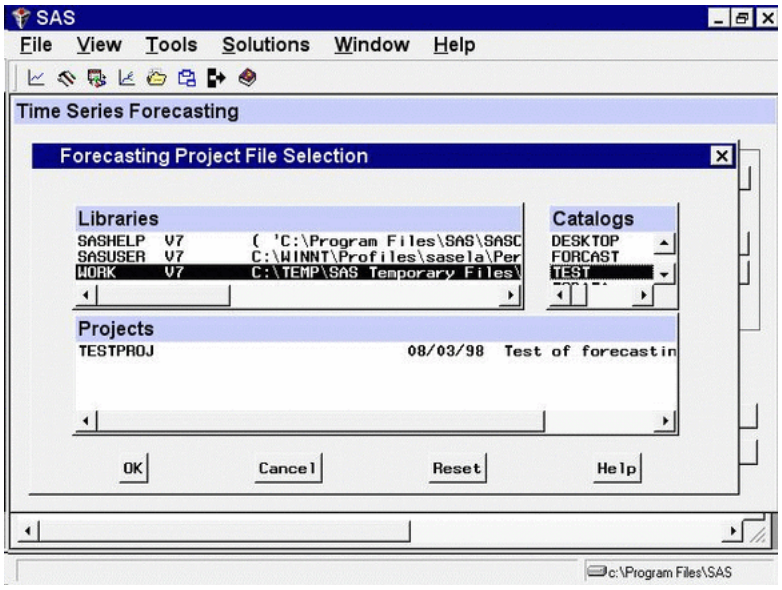
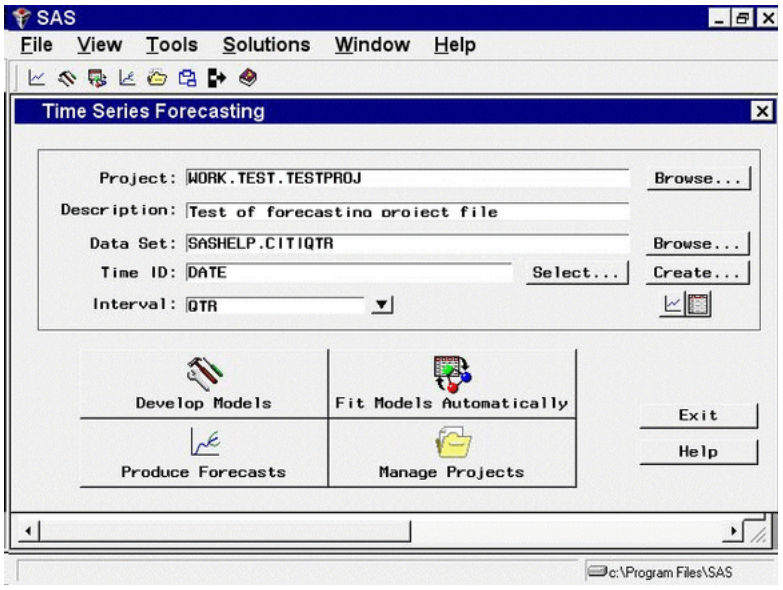

SAS
File View Tools Solutions Window Help
Time Series Forecasting
Forecasting Project File Selection
Libraries
SASHELP V7 ('C:\Program Files\SAS\SASC
SASUSER V7 C:\WINNT\Profiles\sasela\Per
WORK V7 C:\TEMP\SAS Temporary Files\
Catalogs
DESKTOP
FORCAST
TEST
Projects
TESTPROJ 08/03/98 Test of forecastin
OK
Cancel
Reset
Help
c:\Program Files\SAS

Figure 34.27. Forecasting Projects List

Select TESTPROJ from the Projects list and then select the OK button. This returns you to the Time Series Forecasting window.

The system loads the project information you saved in TESTPROJ and displays a message indicating this. The Project field is now set to WORK.TEST.TESTPROJ, and the description is the description you previously gave to TESTPROJ, as shown in Figure 34.28.



Project: WORK.TEST.TESTPROJ
Description: Test of forecasting project file
Data Set: SASHELP.CITIQTR
Time ID: DATE
Interval: QTR

Figure 34.28. Time Series Forecasting Window after Loading Project

If you now select the Manage Projects button, you will see the list of series and forecasting models you created in the previous forecasting session.

Sharing Projects

If you plan to work with others on a forecasting project, you may need to consider how project information can be shared. The series, models, and results of your project are stored in a forecasting project (fmsproj) catalog entry in the location you specify, as illustrated in the previous section. You need only read access to the catalog to work with it, but you must have write access to save the project. Multiple users cannot open a project for update at the same time, but they can do so at different times if they all have write access to the catalog where it is stored.

Project options settings such as the *model selection criterion* and *number of models to keep* are stored in an slist catalog entry in the SASUSER or TSFSUSER library. Write access to this catalog is required. If you have only read access to the SASUSER library, you can use the -RSASUSER option when starting SAS. You will be prompted for a location for the TSFSUSER library, if it is not already assigned. If you want to use TSFSUSER routinely, assign it before you start the Time Series Forecasting System. Select New from the SAS Explorer file menu. In the New Library window, type TSFSUSER for the name. Click the browse button and select the directory or folder you want to use. Turn on the *enable at startup* option so this library will be assigned automatically in subsequent sessions.

The SASUSER library is typically used for private settings saved by individual users. This is the default location for project options. If a work group shares a single options catalog (SASUSER or TSFSUSER points to the same location for all users), then only one user can use the system at a time.

Develop Models Window

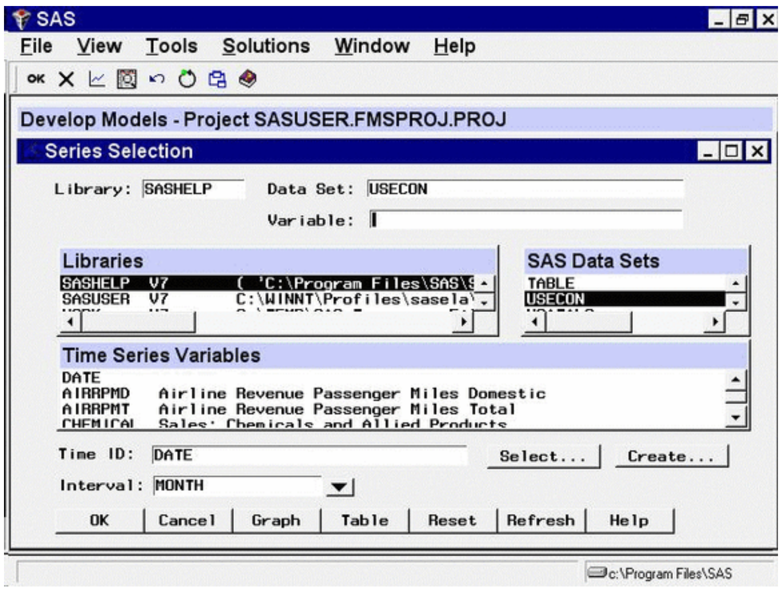
In the first forecasting example, you used the Automatic Model Fitting window to fit and select the forecasting model for each series automatically. In addition to this automatic forecasting process, you can also work with time series one at a time to fit forecasting models and apply your own judgment to choose the best forecasting model for each series.

Using the Automatic Model Fitting feature, the system acts like a "black box." This section goes inside the black box to look at the kinds of forecasting methods that the system provides and introduces some of the tools the system offers to help you find the best forecasting model.

Introduction

From the Time Series Forecasting window, select the Browse button to the right of the Data Set field to bring up the Data Set Selection window. Select the USECON data set from the SASHELP library. This data set contains monthly data on the U.S. economy.

Select OK to close the selection window. Now select the Develop Models button. This brings up the Series Selection window, as shown in Figure 34.29. You can enlarge this window for easier viewing of lists of data sets and series.



Develop Models - Project SASUSER.FMSPROJ.PROJ

Series Selection

Library: SASHELP Data Set: USECON

Variable:

Libraries

```
SASHELP   V7      ( 'C:\Program Files\SAS\
SASUSER   V7      C:\WINNT\Profiles\sasela
```

SAS Data Sets

```
TABLE
USECON
```

Time Series Variables

```
DATE
AIRRPMD   Airline Revenue Passenger Miles Domestic
AIRRPMT   Airline Revenue Passenger Miles Total
CHEMICAL  Sales: Chemicals and Allied Products
```

Time ID: DATE

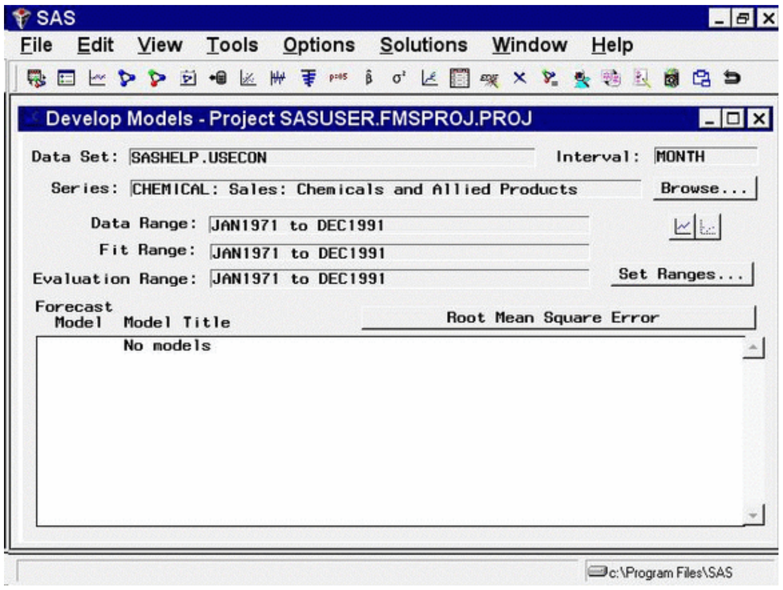
Interval: MONTH

c:\Program Files\SAS

Figure 34.29. Series Selection Window

Select the series CHEMICAL, “Sales of Chemicals and Allied Products”, and then select the OK button.

This brings up the Develop Models window, as shown in Figure 34.30.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
c:\Program Files\SAS

Data Set: SASHELP.USECON
Interval: MONTH
Series: CHEMICAL: Sales: Chemicals and Allied Products
Data Range: JAN1971 to DEC1991
Fit Range: JAN1971 to DEC1991
Evaluation Range: JAN1971 to DEC1991

Forecast Model	Model Title	Root Mean Square Error
	No models	

Figure 34.30. Develop Models Window

The Data Set, Interval, and Series fields in the upper part of the Develop Models window indicate the series with which you are currently working. You can change the settings of these fields by selecting the Browse button.

The Data Range, Fit Range, and Evaluation Range fields show the time period over which data are available for the current series, and what parts of that time period will be used to fit forecasting models to the series and to evaluate how well the models fit the data. You can change the settings of these fields by selecting the Set Ranges button.

The bottom part of the Develop Models window consists of a table of forecasting models fit to the series. Initially, the list is empty, as indicated by the message "No models." You can fit any number of forecasting models to each series and designate which one you want to use to produce forecasts.

Graphical tools are available for exploring time series and fitted models. The two icons below the Browse button access the `Time Series Viewer` and the Model Viewer.

Select the left icon. This brings up the Time Series Viewer window, as shown in Figure 34.31.

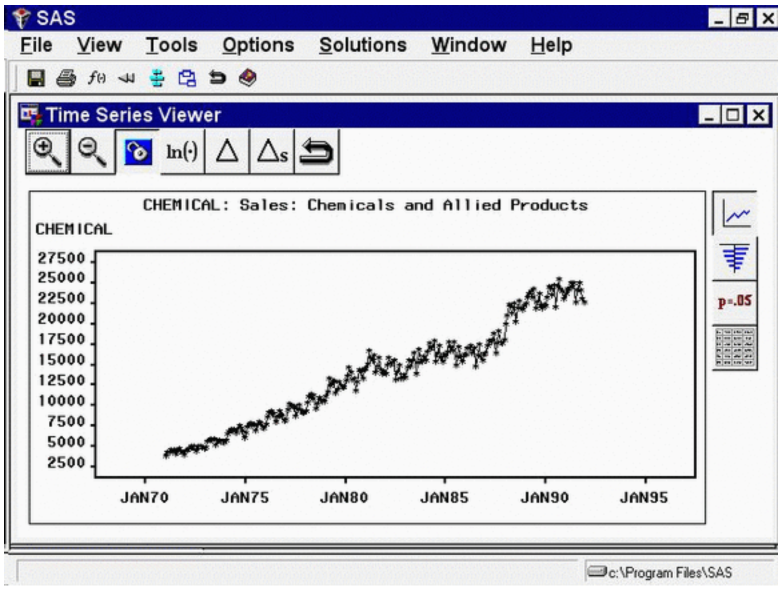


Figure 34.31. Chemical and Allied Product Series

The Time Series Viewer displays a plot of the CHEMICAL series. The Time Series Viewer offers many useful features, which are explored in later sections.

The Time Series Viewer appears in a separate resizable window. You can switch back and forth between the Time Series Viewer window and other windows. For now, return to the Develop Models window. You can close the Time Series Viewer window or leave it open. (To close the Time Series Viewer window, select `Close` from the toolbar or from the File pull-down menu.)

Fitting Models

To bring up a menu of model fitting choices, select `Edit` from the menu bar and then select `Fit Model`, or select `Fit Models from List` in the tool-bar, or simply select a blank line in the table as shown in Figure 34.32.

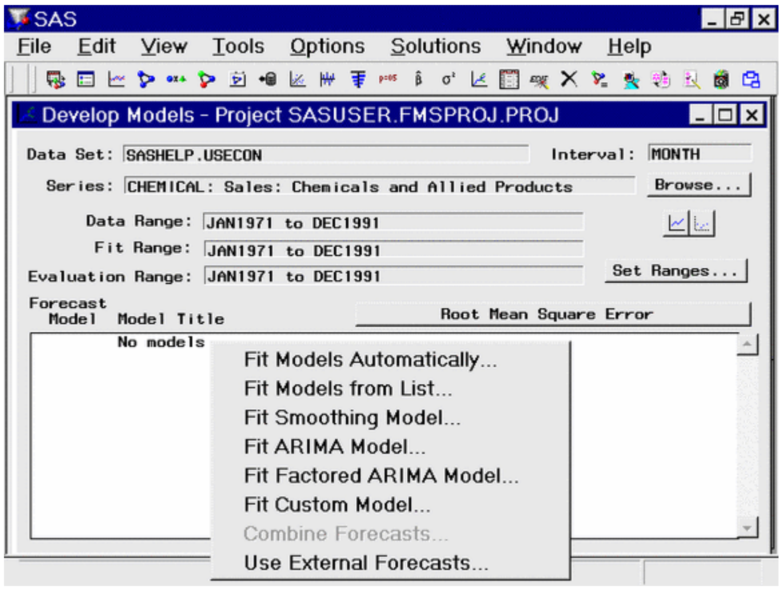

SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Data Set: SASHELP.USECON
Interval: MONTH
Series: CHEMICAL: Sales: Chemicals and Allied Products
Browse...
Data Range: JAN1971 to DEC1991
Fit Range: JAN1971 to DEC1991
Evaluation Range: JAN1971 to DEC1991
Set Ranges...
Forecast Model
Model Title
Root Mean Square Error
No models
Fit Models Automatically...
Fit Models from List...
Fit Smoothing Model...
Fit ARIMA Model...
Fit Factored ARIMA Model...
Fit Custom Model...
Combine Forecasts...
Use External Forecasts...

Figure 34.32. Menu of Model Fitting Choices

The Forecasting System provides several ways to specify forecasting models. The eight choices given by the menu shown in Figure 34.32 are as follows:

`Fit Models Automatically`
: performs for the current series the same automatic model selection process that the Automatic Model Fitting window applies to a set of series.

`Fit Models from List`
: presents a list of commonly used forecasting models for convenient point-and-click selection.

`Fit Smoothing Model`
: displays the Smoothing Model Specification window, which enables you to specify several kinds of exponential smoothing and Winters method forecasting models.

`Fit ARIMA Model`
: displays the ARIMA Model Specification window, which enables you to specify many kinds of autoregressive integrated moving average (ARIMA) models, including seasonal ARIMA models and ARIMA models with regressors, transfer functions, and other predictors.

`Fit Factored ARIMA Model`
: displays the Factored ARIMA Model Specification window, which enables you to specify more general ARIMA models, including subset models and models with unusual and/or multiple seasonal cycles. It also supports regressors, transfer functions, and other predictors.

`Fit Custom Model`
: displays the Custom Model Specification window, which enables you to construct a forecasting model by specifying separate options for transforming the data, modeling the trend, modeling seasonality, modeling autocorrelation of the errors, and modeling the effect of regressors and other independent predictors.

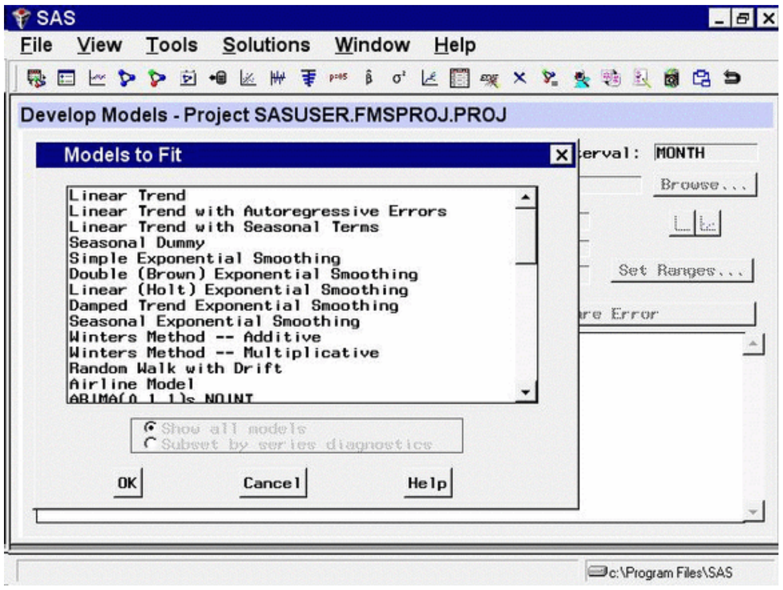
`Combine Forecasts`
: displays the Forecast Combination Model Specification window, which enables you to specify models that produce forecasts by combining, or averaging, the forecasts from other models. (This option is not available unless you have fit at least two models.)

`Use External Forecasts`
: displays the External Forecast Model Specification window, which enables you to use judgmental or externally produced forecasts that have been saved in a separate series in the data set.

All of the forecasting models used by the system are ultimately specified through one of the four windows: Smoothing Method Specification, ARIMA Model Specification, Factored ARIMA Model Specification, or Custom Model Specification. You can specify the same models with either the ARIMA Model Specification window or the Custom Model Specification window, but the Custom Model Specification window may provide a more natural way to specify models for those who are less familiar with the Box-Jenkins style of time series model specification.

The Automatic Model feature, the Models to Fit window, and the Forecast Combination Model Specification window all deal with lists of forecasting models previously defined through the Smoothing Model, ARIMA Model, or Custom Model specification windows. These windows are discussed in detail in later sections.

To get started using the Develop Models window, select the Fit Models from List item from the menu shown in Figure 34.32. This brings up the Models to Fit window, as shown in Figure 34.33.



Develop Models - Project SASUSER.FMSPROJ.PROJ

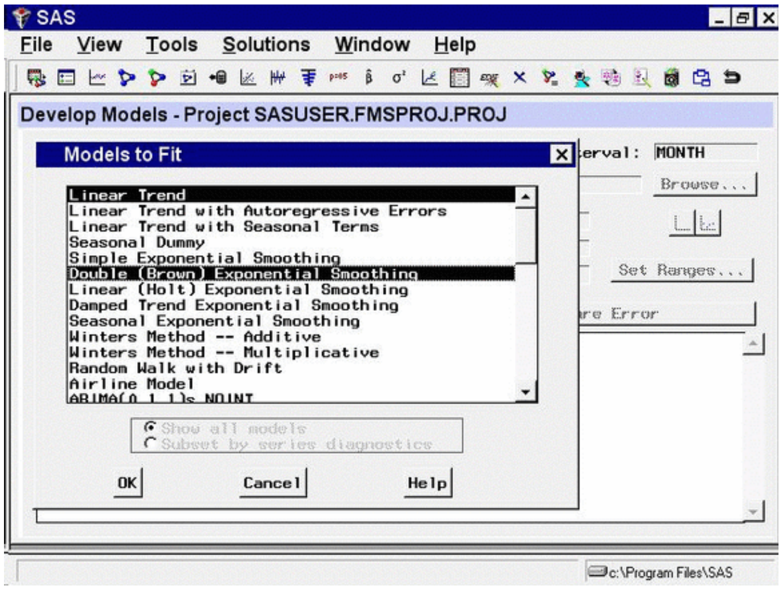
Models to Fit

```
Linear Trend
Linear Trend with Autoregressive Errors
Linear Trend with Seasonal Terms
Seasonal Dummy
Simple Exponential Smoothing
Double (Brown) Exponential Smoothing
Linear (Holt) Exponential Smoothing
Damped Trend Exponential Smoothing
Seasonal Exponential Smoothing
Winters Method -- Additive
Winters Method -- Multiplicative
Random Walk with Drift
Airline Model
ARIMA(0,1,1)s NOINT
```

(●) Show all models
( ) Subset by series diagnostics

Figure 34.33. Models to Fit Window

You can select several models to fit at once by holding down the CTRL key as you make the selections. Select `Linear Trend` and `Double (Brown) Exponential Smoothing`, as shown in Figure 34.34, and then select the OK button.



Develop Models - Project SASUSER.FMSPROJ.PROJ

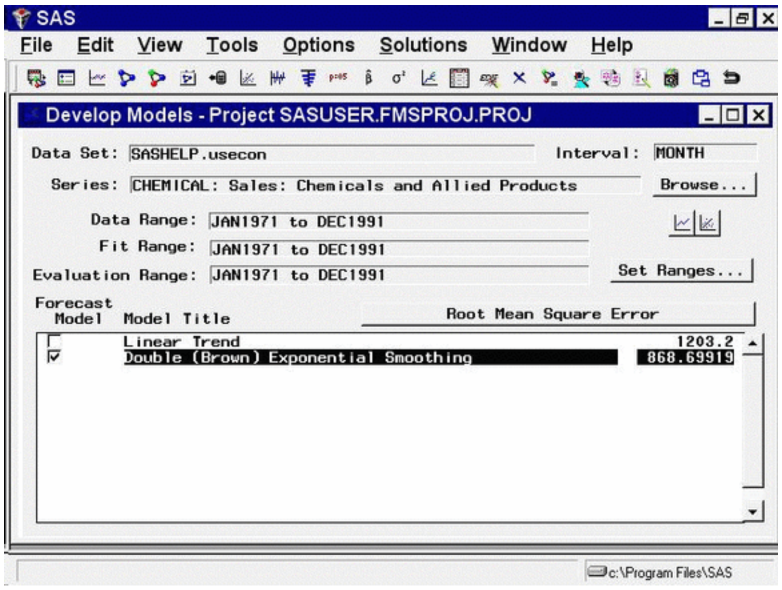
Models to Fit

```
Linear Trend
Linear Trend with Autoregressive Errors
Linear Trend with Seasonal Terms
Seasonal Dummy
Simple Exponential Smoothing
Double (Brown) Exponential Smoothing
Linear (Holt) Exponential Smoothing
Damped Trend Exponential Smoothing
Seasonal Exponential Smoothing
Winters Method -- Additive
Winters Method -- Multiplicative
Random Walk with Drift
Airline Model
ARIMA(0,1,1)s NOINT
```

(●) Show all models
( ) Subset by series diagnostics

Figure 34.34. Selecting Models to Fit

The system fits the two models you selected. After the models are fit, the labels of the two models and their goodness-of-fit statistic are added to the model table, as shown in Figure 34.35.



SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ

Data Set: SASHELP.usecon Interval: MONTH
Series: CHEMICAL: Sales: Chemicals and Allied Products Browse...
Data Range: JAN1971 to DEC1991
Fit Range: JAN1971 to DEC1991
Evaluation Range: JAN1971 to DEC1991 Set Ranges...

Forecast Model	Model Title	Root Mean Square Error
☐	Linear Trend	1203.2
☑	Double (Brown) Exponential Smoothing	868.69919

Figure 34.35. Fitted Models List

Model List and Statistics of Fit

In the model list, the *Model Title* column shows the descriptive labels for the two fitted models, in this case Linear Trend and Double Exponential Smoothing.

The column labeled *Root Mean Square Error* (or labeled *Mean Absolute Percent Error* if you continued from the example in the previous section) shows the goodness-of-fit criterion used to decide which model fits better. By default, the criterion used is the root mean square error, but you can choose a different measure of fit. The linear trend model has a root mean square error of 1203, while the double exponential smoothing model fits better, with a RMSE of only 869.

The left column labeled *Forecast Model* consists of check boxes that indicate which one of the models in the list has been selected as the model to use to produce the forecasts for the series. When new models are fit and added to the model list, the system sets the Forecast Model flags to designate the one model with the best fit–as measured by the selected goodness-of-fit statistic–as the forecast model. (In the case of ties, the first model with the best fit is selected.)

Because the Double Exponential Smoothing model has the smaller RMSE of the two models in the list, its Forecast Model check box is set. If you would rather produce forecasts using the Linear Trend model, choose it by selecting the corresponding check box in the Forecast Model column.

To use a different goodness-of-fit criterion, select the button with the current criterion name on it (Root Mean Square Error or Mean Absolute Percent Error). This brings up the Model Selection Criterion window, as shown in Figure 34.36.

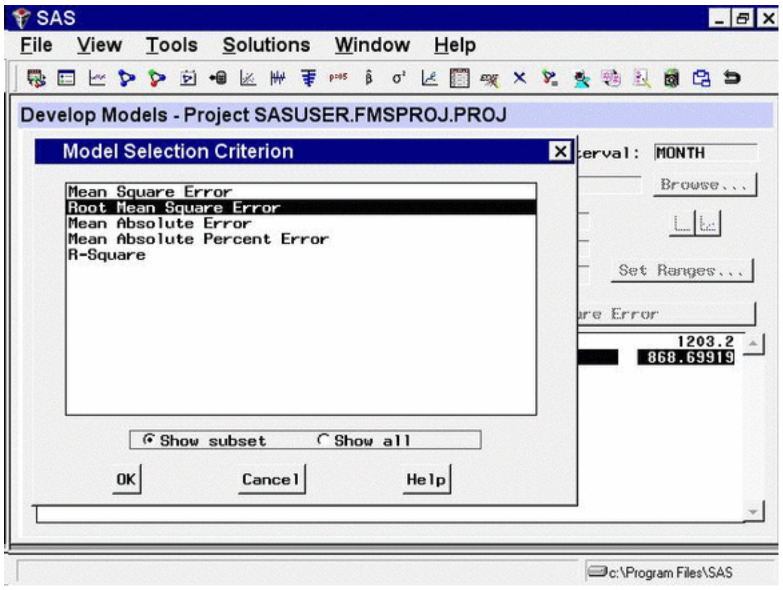


Figure 34.36. Model Selection Criterion Window

The system provides many measures of fit that you can use as the model selection criterion. To avoid confusion, only the most popular of the available fit statistics are shown in this window by default. To display the complete list, you can select the `Show all` option. You can control the subset of statistics listed in this window through the Statistics of Fit item in the Options menu on the Develop Models window.

Initially, Root Mean Square Error is selected. Select R-Square and then select the OK button. This changes the fit statistic displayed in the model list, as shown in Figure 34.37.

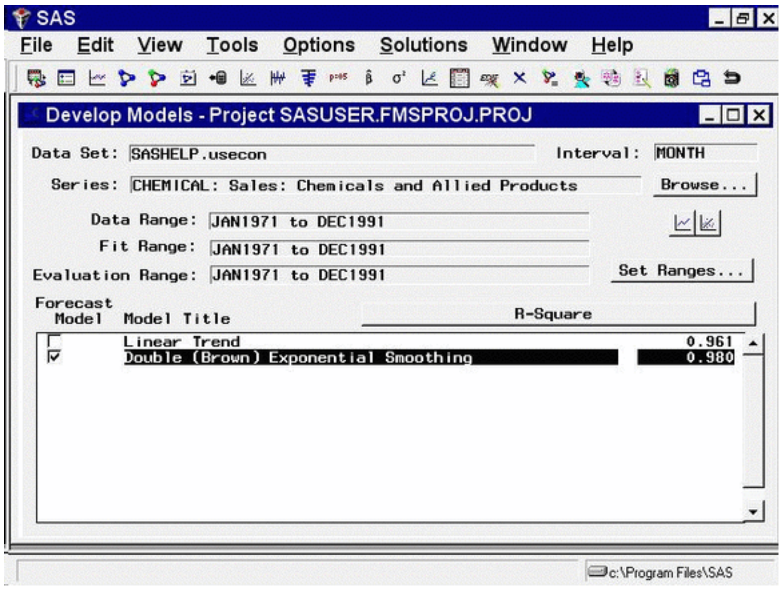
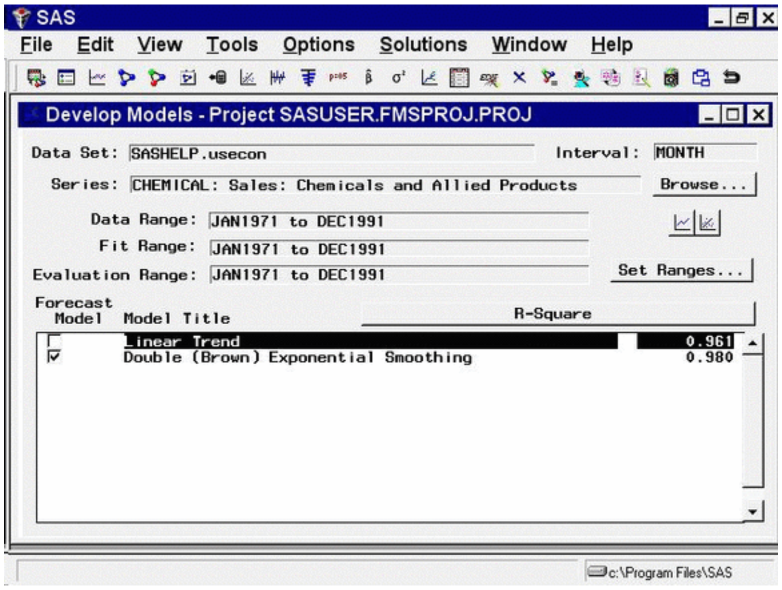

SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Data Set: SASHELP.usecon
Interval: MONTH
Series: CHEMICAL: Sales: Chemicals and Allied Products
Browse...
Data Range: JAN1971 to DEC1991
Fit Range: JAN1971 to DEC1991
Evaluation Range: JAN1971 to DEC1991
Set Ranges...
Forecast Model
Model Title
R-Square
Linear Trend
0.961
Double (Brown) Exponential Smoothing
0.980
c:\Program Files\SAS

Figure 34.37. Model List with R-Square Statistics

Now that you have fit some models to the series, you can use the Model Viewer button to take a closer look at the predictions of these models.

Model Viewer

In the Develop Models window, select the row in the table containing the Linear Trend model so that this model is highlighted. The model list should now appear as shown in Figure 34.38.



Develop Models - Project SASUSER.FMSPROJ.PROJ

Data Set: SASHELP.usecon Interval: MONTH

Series: CHEMICAL: Sales: Chemicals and Allied Products

Data Range: JAN1971 to DEC1991

Fit Range: JAN1971 to DEC1991

Evaluation Range: JAN1971 to DEC1991

Forecast Model	Model Title	R-Square
☐	Linear Trend	0.961
✓	Double (Brown) Exponential Smoothing	0.980

Figure 34.38. Selecting a Model to View

Note that the Linear Trend model is now highlighted, but the Forecast Model column still shows the Double Exponential Smoothing model as the model chosen to produce the final forecasts for the series. Selecting a model in the list means that this is the model that menu items such as `View Model`, `Delete`, `Edit`, and `Refit` will act upon. Choosing a model by selecting its check box in the Forecast Model column means that this model will be used by the Produce Forecasts process to generate forecasts.

Now bring up the Model Viewer by selecting the right icon under the Browse button, or by selecting `Model Predictions` in the tool-bar or from the View pull-down menu. The Model Viewer displays the Linear Trend model, as shown in Figure 34.39.

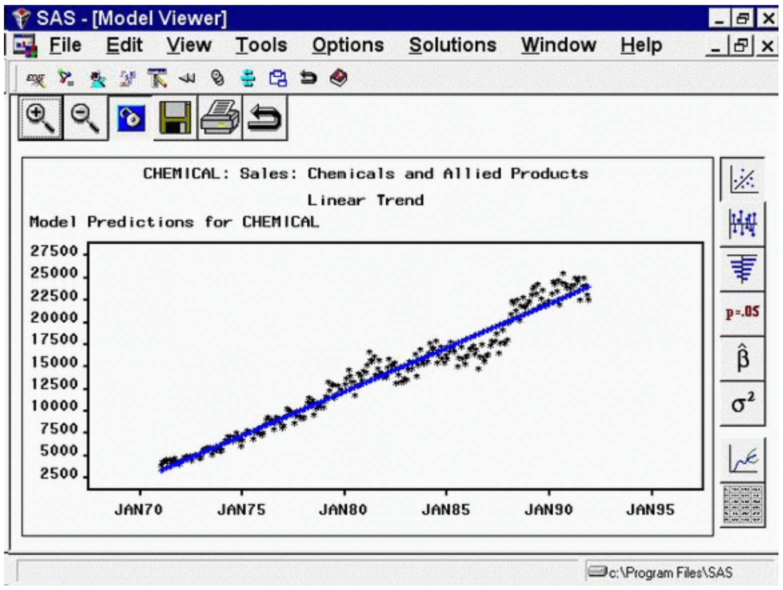

SAS - [Model Viewer]
File Edit View Tools Options Solutions Window Help
CHEMICAL: Sales: Chemicals and Allied Products
Linear Trend
Model Predictions for CHEMICAL
27500
25000
22500
20000
17500
15000
12500
10000
7500
5000
2500
JAN70 JAN75 JAN80 JAN85 JAN90 JAN95
p=.05
c:\Program Files\SAS

Figure 34.39. Model Viewer: Actual and Predicted Values Plot

This graph shows the linear trend line representing the model predicted values together with a plot of the actual data values, which fluctuate about the trend line.

Prediction Error Plots

Select the second icon from the top in the vertical tool-bar in the Model Viewer window. This switches the Viewer to display a plot of the model prediction errors (actual data values minus the predicted values), as shown in Figure 34.40.

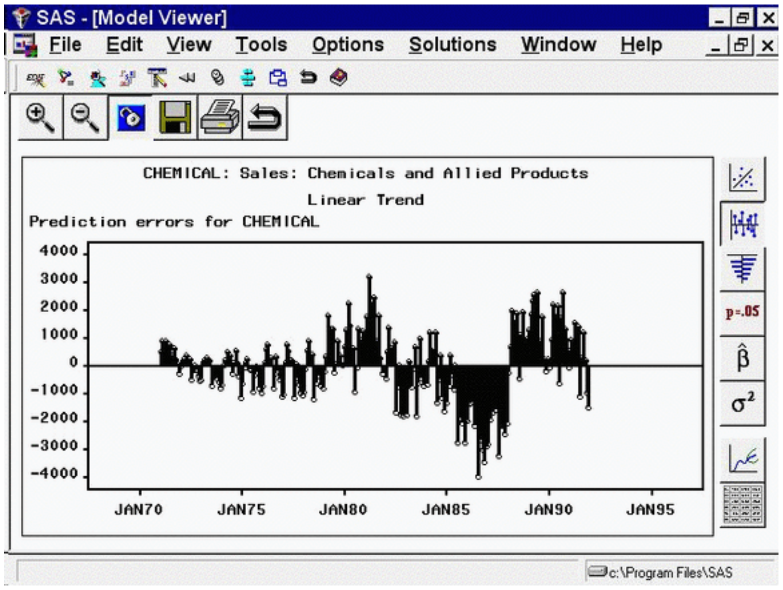

SAS - [Model Viewer]
File Edit View Tools Options Solutions Window Help
CHEMICAL: Sales: Chemicals and Allied Products
Linear Trend
Prediction errors for CHEMICAL
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
JAN70 JAN75 JAN80 JAN85 JAN90 JAN95
p=.05
c:\Program Files\SAS

Figure 34.40. Model Viewer: Prediction Errors Plot

If the model being viewed includes a transformation, prediction errors are defined as the difference between the transformed series actual values and model predictions. You can choose to graph instead the difference between the untransformed series values and untransformed model predictions, which are called *model residuals*. You can also graph normalized prediction errors or normalized model residuals. Use the Residual Plot Options submenu under the Options pull-down menu.

Autocorrelation Plots

Select the third icon from the top in the vertical tool-bar. This switches the Viewer to display a plot of autocorrelations of the model prediction errors at different lags, as shown in Figure 34.41. Autocorrelations, partial autocorrelations, and inverse autocorrelations are displayed, with lines overlaid at plus and minus two standard errors. You can switch the graphs so that the bars represent significance probabilities by selecting the Correlation Probabilities item on the tool-bar or from the View pull-down menu. For more information on the meaning and use of autocorrelation plots, refer to Chapter 11, "The ARIMA Procedure."

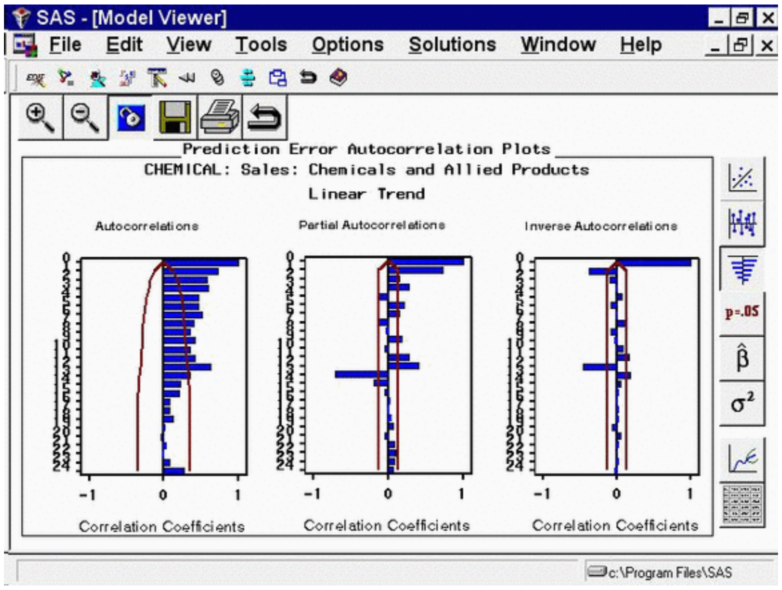


Figure 34.41. Model Viewer: Autocorrelations Plot

White Noise and Stationarity Plots

Select the fourth icon from the top in the vertical tool-bar. This switches the Viewer to display a plot of white noise and stationarity tests on the model prediction errors, as shown in Figure 34.42.

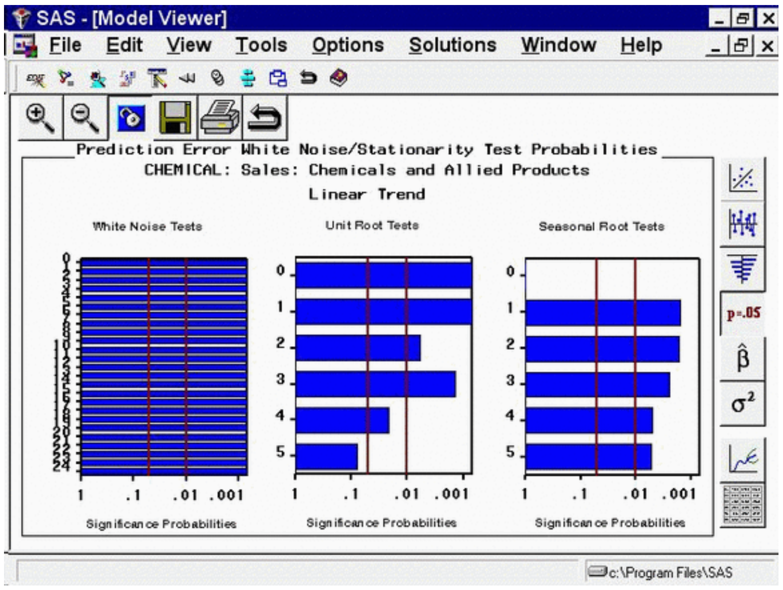
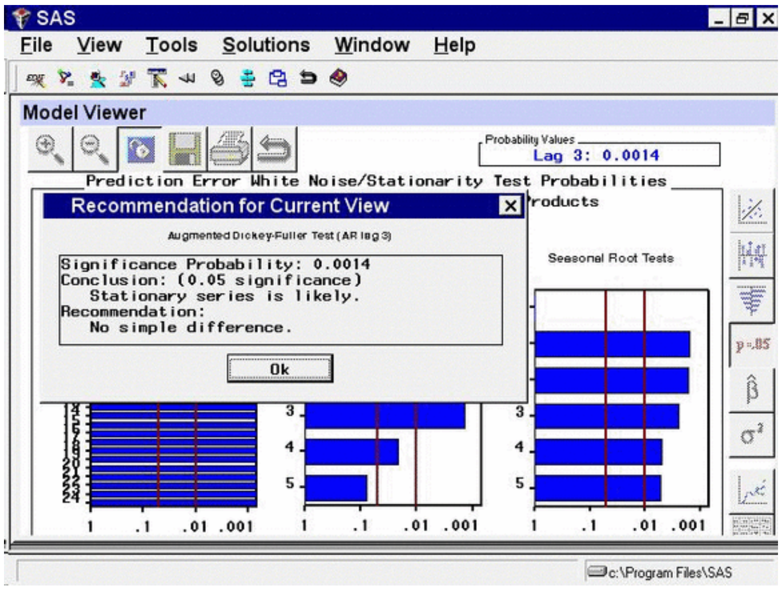

SAS - [Model Viewer]
File Edit View Tools Options Solutions Window Help
Prediction Error White Noise/Stationarity Test Probabilities
CHEMICAL: Sales: Chemicals and Allied Products
Linear Trend
White Noise Tests
Unit Root Tests
Seasonal Root Tests
Significance Probabilities
1 .1 .01 .001
p=.05
c:\Program Files\SAS

Figure 34.42. Model Viewer: White Noise and Stationarity Plot

The white noise test bar chart shows significance probabilities of the Ljung-Box Chi Square statistic. Each bar shows the probability computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the prediction errors represent white noise. In this example they are all significant beyond the .001 probability level, so that we reject the null hypothesis. In other words, the high level of significance at all lags makes it clear that the linear trend model is inadequate for this series.

The second bar chart shows significance probabilities of the Augmented Dickey-Fuller test for unit roots. For example, the bar at lag three indicates a probability of .0014, so that we reject the null hypothesis that the series is nonstationary. The third bar chart is similar to the second except that it represents the seasonal lags. Since this series has a yearly seasonal cycle, the bars represent yearly intervals.

You can select any of the bars to display an interpretation. Select the fourth bar of the middle chart. This displays the `Recommendation for Current View`, as shown in Figure 34.43. This window gives an interpretation of the test represented by the bar that was selected; it is significant, therefore a stationary series is likely. It also gives a recommendation: You do not need to perform a simple difference to make the series stationary.



Probability Values
Lag 3: 0.0014

Prediction Error White Noise/Stationarity Test Probabilities

Recommendation for Current View

Augmented Dickey-Fuller Test (AR lag 3)

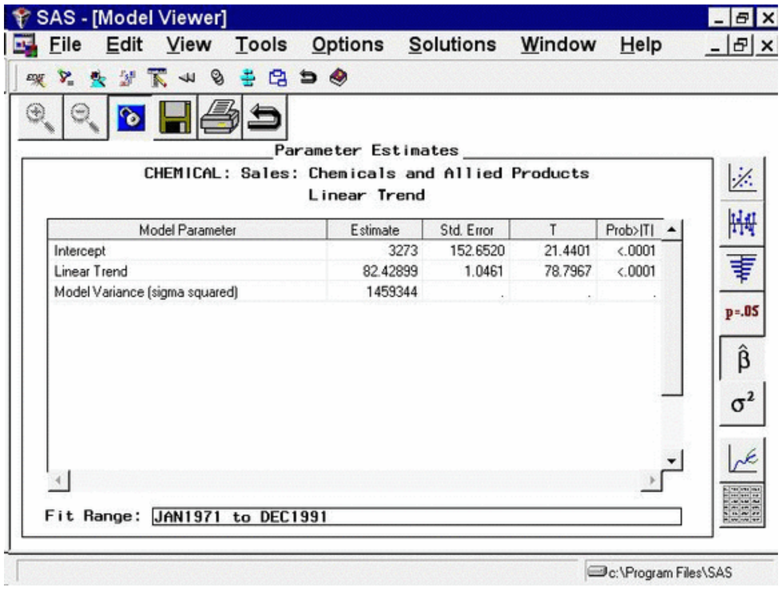
```
Significance Probability: 0.0014
Conclusion: (0.05 significance)
   Stationary series is likely.
Recommendation:
   No simple difference.
```

Ok

Figure 34.43. Model Viewer: Recommendation for Current View

Parameter Estimates Table

Select the fifth icon from the top in the vertical tool-bar to the right of the graph. This switches the Viewer to display a table of parameter estimates for the fitted model, as shown in Figure 34.44.



Parameter Estimates

CHEMICAL: Sales: Chemicals and Allied Products
Linear Trend

Model Parameter	Estimate	Std. Error	T	Prob>|T|
Intercept	3273	152.6520	21.4401	<.0001
Linear Trend	82.42899	1.0461	78.7967	<.0001
Model Variance (sigma squared)	1459344	.	.	.

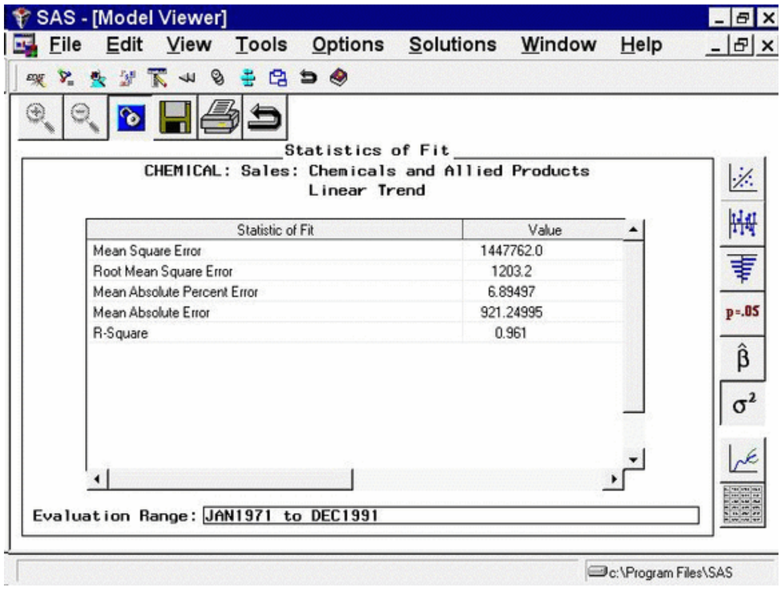
Fit Range: JAN1971 to DEC1991

Figure 34.44. Model Viewer: Parameter Estimates Table

For the linear trend model, the parameters are the intercept and slope coefficients. The table shows the values of the fitted coefficients together with standard errors and *t*-tests for the statistical significance of the estimates. The model residual variance is also shown.

Statistics of Fit Table

Select the sixth icon from the top in the vertical tool-bar to the right of the table. This switches the Viewer to display a table of statistics of fit computed from the model prediction errors, as shown in Figure 34.45. The list of statistics displayed is controlled by selecting `Statistics of Fit` from the Options pull-down menu.



Statistics of Fit
CHEMICAL: Sales: Chemicals and Allied Products
Linear Trend

Statistic of Fit	Value
Mean Square Error	1447762.0
Root Mean Square Error	1203.2
Mean Absolute Percent Error	6.89497
Mean Absolute Error	921.24995
R-Square	0.961

Evaluation Range: JAN1971 to DEC1991

Figure 34.45. Model Viewer: Statistics of Fit Table

Changing to a Different Model

Select the first icon in the vertical tool-bar to the right of the table to return the display to the predicted and actual values plots (Figure 34.39).

Now return to the Develop Models window, but do not close the Model Viewer window. You can use the Next Viewer icon in the tool-bar or your system's window manager controls to switch windows. You can resize the windows to make them both visible.

Select the Double Exponential Smoothing model so that this line of the model list is highlighted. The Model Viewer window is now updated to display a plot of the predicted values for the Double Exponential Smoothing model, as shown in Figure 34.46. The Model Viewer is automatically updated to display the currently selected model, unless you specify `Unlink` (the third icon in the window's horizontal tool-bar).

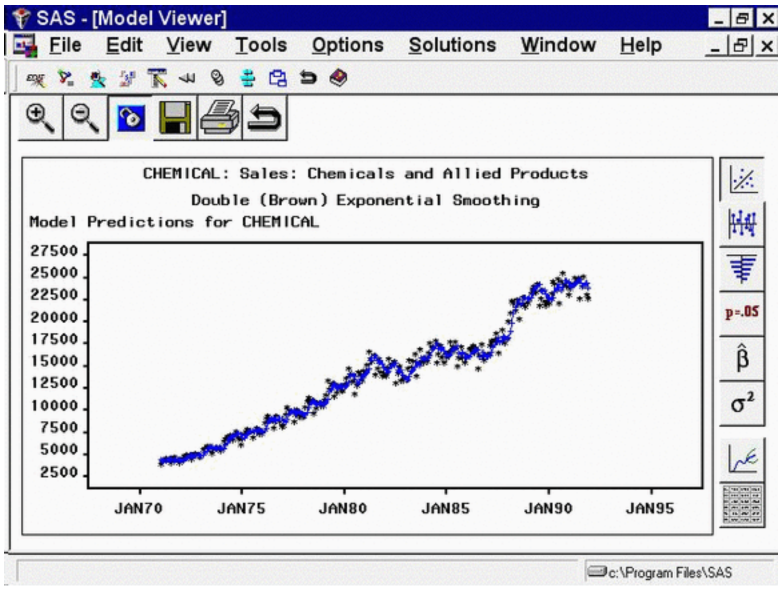

SAS - [Model Viewer]
File Edit View Tools Options Solutions Window Help
CHEMICAL: Sales: Chemicals and Allied Products
Double (Brown) Exponential Smoothing
Model Predictions for CHEMICAL
27500
25000
22500
20000
17500
15000
12500
10000
7500
5000
2500
JAN70
JAN75
JAN80
JAN85
JAN90
JAN95
p=.05
c:\Program Files\SAS

Figure 34.46. Model Viewer Plot for Exponential Smoothing Model

Forecasts and Confidence Limits Plots

Select the seventh icon from the top in the vertical tool-bar to the right of the graph. This switches the Viewer to display a plot of forecast values and confidence limits, together with actual values and one-step-ahead within-sample predictions, as shown in Figure 34.47.

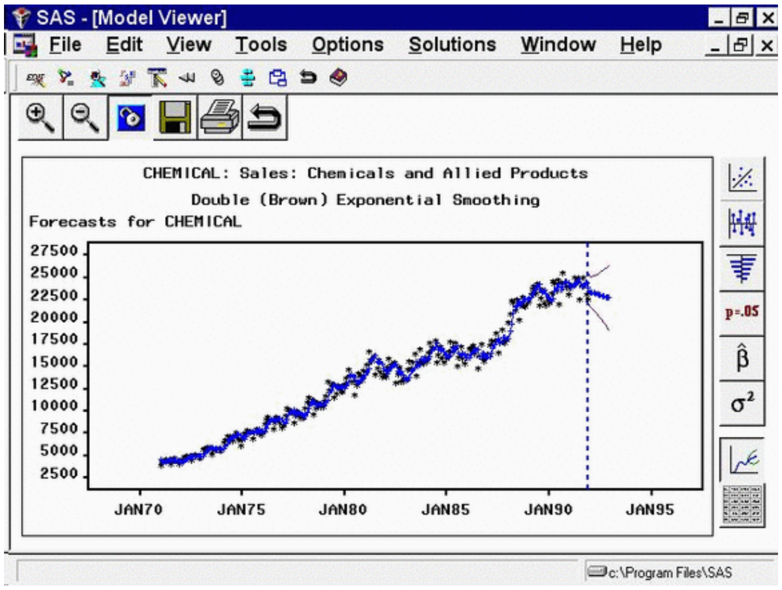
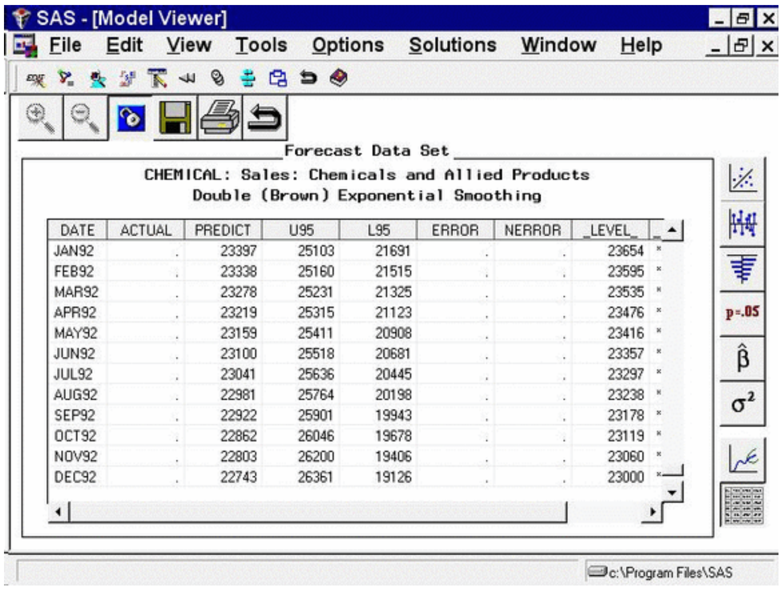


Figure 34.47. Model Viewer: Forecasts and Confidence Limits

Data Table

Select the last icon at the bottom of the vertical tool-bar to the right of the graph. This switches the Viewer to display the forecast data set as a table, as shown in Figure 34.48.



Forecast Data Set

CHEMICAL: Sales: Chemicals and Allied Products
Double (Brown) Exponential Smoothing

DATE	ACTUAL	PREDICT	U95	L95	ERROR	NERROR	_LEVEL_	_
JAN92	.	23397	25103	21691	.	.	23654	*
FEB92	.	23338	25160	21515	.	.	23595	*
MAR92	.	23278	25231	21325	.	.	23535	*
APR92	.	23219	25315	21123	.	.	23476	*
MAY92	.	23159	25411	20908	.	.	23416	*
JUN92	.	23100	25518	20681	.	.	23357	*
JUL92	.	23041	25636	20445	.	.	23297	*
AUG92	.	22981	25764	20198	.	.	23238	*
SEP92	.	22922	25901	19943	.	.	23178	*
OCT92	.	22862	26046	19678	.	.	23119	*
NOV92	.	22803	26200	19406	.	.	23060	*
DEC92	.	22743	26361	19126	.	.	23000	*

Figure 34.48. Model Viewer: Forecast Data Table

To view the full data set, use the vertical and horizontal scroll bars on the data table or enlarge the window.

Closing the Model Viewer

Other features of the Model Viewer and Develop Models window are discussed later in this book. For now, close the Model Viewer window and return to the Time Series Forecasting window.

To close the Model Viewer window, select `Close` from the window's horizontal tool-bar or from the File pull-down menu.

Chapter 35
Creating Time ID Variables

Chapter Contents

Chapter 35

Creating Time ID Variables

The Forecasting System requires that the input data set contain a time ID variable. If the data you want to forecast are not in this form, you can use features of the Forecasting System to help you add time ID variables to your data set. This chapter shows examples of how to use these features.

Creating a Time ID Value from a Starting Date and Frequency

As a first example of adding a time ID variable, you will use the SAS data set created by the following statements. (Or use your own data set if you prefer.)

```
data no_id;
  input y @@;
datalines;
  10 15 20 25 30 35 40 45
  50 55 60 65 70 75 80 85
run;
```

Submit these SAS statements to create the data set NO_ID. This data set contains the single variable Y. Assume that Y is a quarterly series and starts in the first quarter of 1991.

In the `Time Series Forecasting` window, use the Browse button to the right of the `Data set` field to bring up the `Data Set Selection` window. Select the WORK library and then select the NO_ID data set.

You must create a time ID variable for the data set. Click the Create button to the right of the Time ID field. This brings up a menu of choices for creating the Time ID variable, as shown in Figure 35.1.

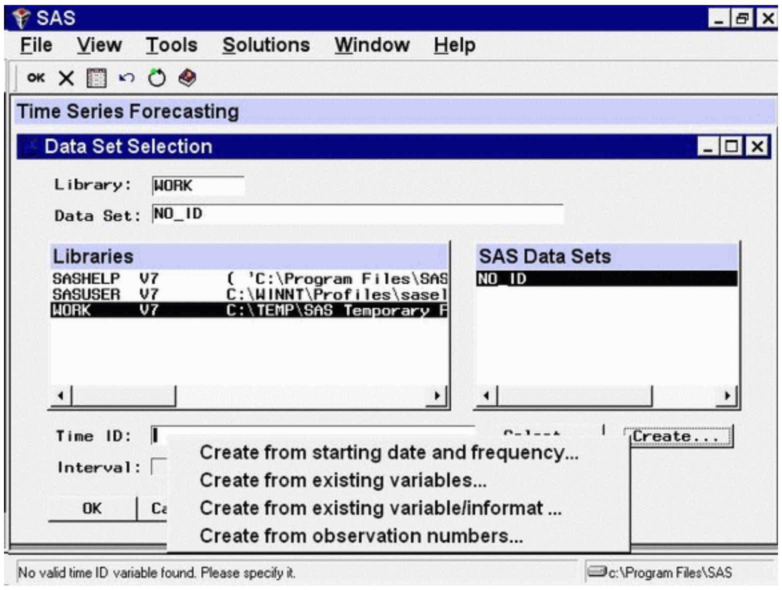

SAS
File View Tools Solutions Window Help
Time Series Forecasting
Data Set Selection
Library: WORK
Data Set: NO_ID
Libraries
SASHELP V7 ('C:\Program Files\SAS
SASUSER V7 C:\WINNT\Profiles\sasel
WORK V7 C:\TEMP\SAS Temporary F
SAS Data Sets
NO_ID
Time ID:
Interval:
OK
Create...
Create from starting date and frequency...
Create from existing variables...
Create from existing variable/informat ...
Create from observation numbers...
No valid time ID variable found. Please specify it.
c:\Program Files\SAS

Figure 35.1. Time ID Creation Popup Menu

Select the first choice, `Create from starting date and frequency`. This brings up the `Time ID Creation from Starting Date` window shown in Figure 35.2.

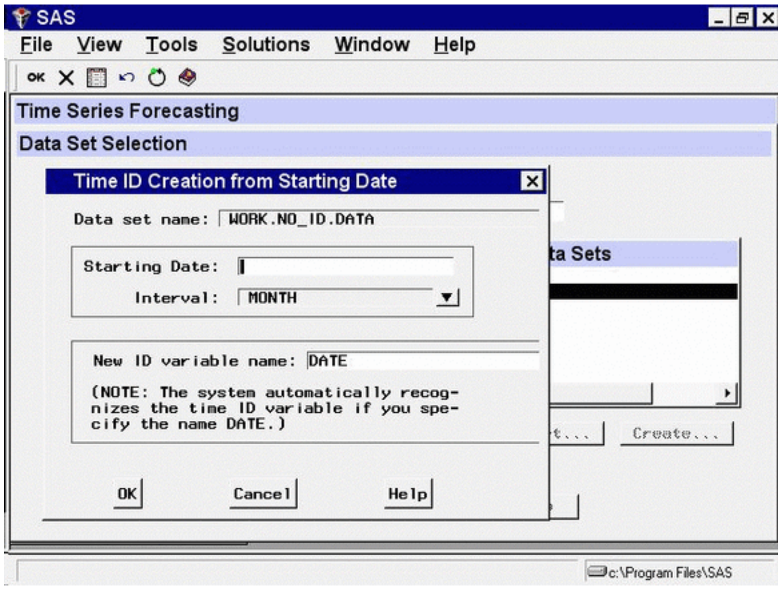

SAS
File View Tools Solutions Window Help
Time Series Forecasting
Data Set Selection
Time ID Creation from Starting Date
Data set name: WORK.NO_ID.DATA
Starting Date:
Interval: MONTH
New ID variable name: DATE
(NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.)
OK
Cancel
Help
Create...
c:\Program Files\SAS

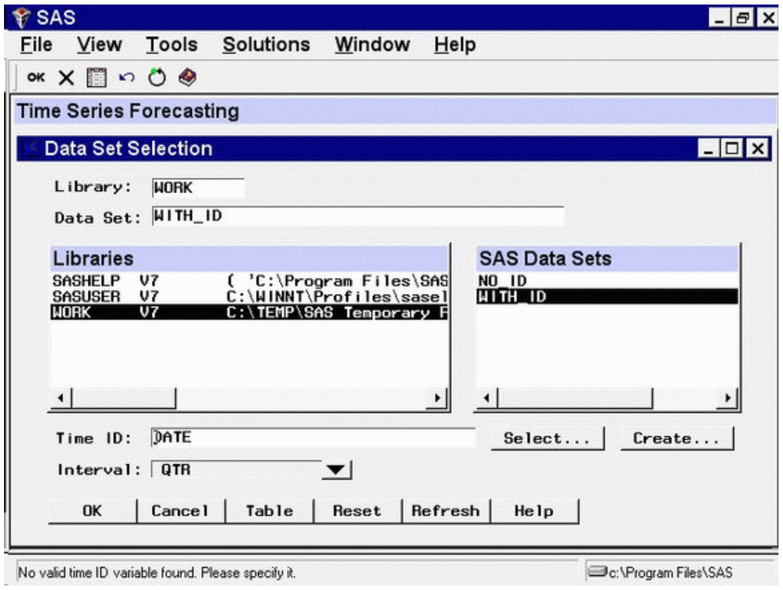
Figure 35.2. Time ID Creation from Starting Date Window

Enter the starting date, 1991:1, in the `Starting Date` field.

Select the `Interval` combo box arrow and select QTR from the pop-up menu. The Interval value QTR means that the time interval between successive observations is a quarter of a year; that is, the data frequency is quarterly.

Now select the `OK` button. The system prompts you for the name of the new data set. If you want to create a new copy of the input data set with the DATE variable added, you should enter a name for the new data set. If you want to replace the NO_ID data set with the new copy containing DATE, just select the `OK` button without changing the name.

For this example, change the `New name` field to WITH_ID and select the `OK` button. The data set WITH_ID is created containing the series Y from NO_ID and the added ID variable DATE. The system returns to the `Data Set Selection` window, which now appears as shown in Figure 35.3.



Library: WORK
Data Set: WITH_ID

Libraries		
SASHELP	V7	('C:\Program Files\SAS
SASUSER	V7	C:\WINNT\Profiles\sasel
WORK	V7	C:\TEMP\SAS Temporary F

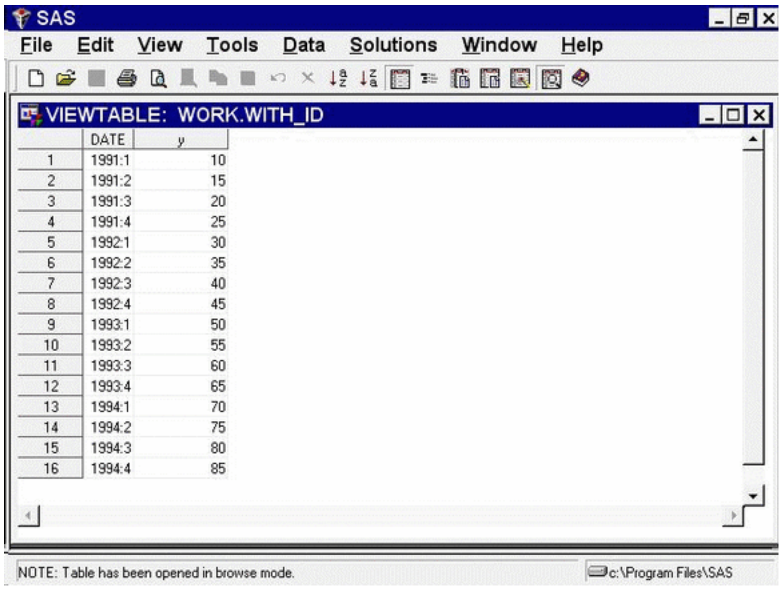
SAS Data Sets: NO_ID, WITH_ID

Time ID: DATE
Interval: QTR

No valid time ID variable found. Please specify it.
c:\Program Files\SAS

Figure 35.3. Data Set Selection Window after Creating Time ID

Select the `Table` button to see the new data set WITH_ID. This brings up a VIEWTABLE window on the data set WITH_ID, as shown in Figure 35.4. Select `File` and `Close` to close the VIEWTABLE window.



	DATE	y
1	1991:1	10
2	1991:2	15
3	1991:3	20
4	1991:4	25
5	1992:1	30
6	1992:2	35
7	1992:3	40
8	1992:4	45
9	1993:1	50
10	1993:2	55
11	1993:3	60
12	1993:4	65
13	1994:1	70
14	1994:2	75
15	1994:3	80
16	1994:4	85

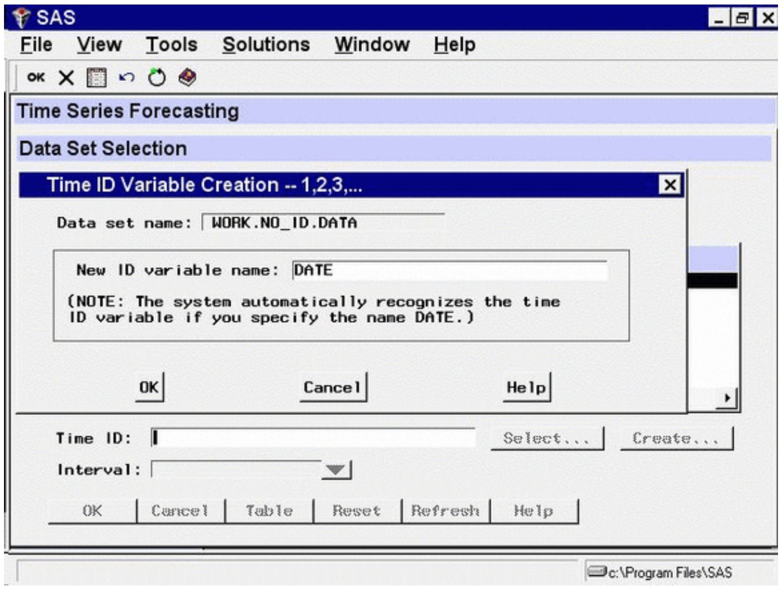
NOTE: Table has been opened in browse mode.
c:\Program Files\SAS

Figure 35.4. Viewtable Display of Data Set with Time ID Added

Using Observation Numbers as the Time ID

Normally, the time ID variable will contain date values. If you do not want to have dates associated with your forecasts, you can also use observation numbers as time ID variables. However, you still must have an ID variable. This can be illustrated by adding an observation index time ID variable to the data set NO_ID.

In the Data Set Selection window, select the data set NO_ID again. Select the Create button to the right of the `Time ID` field. Select the fourth choice, `Create from observation numbers`, from the pop-up menu. This brings up the `Time ID Variable Creation` window shown in Figure 35.5.



Data set name: WORK.NO_ID.DATA

New ID variable name: DATE

(NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.)

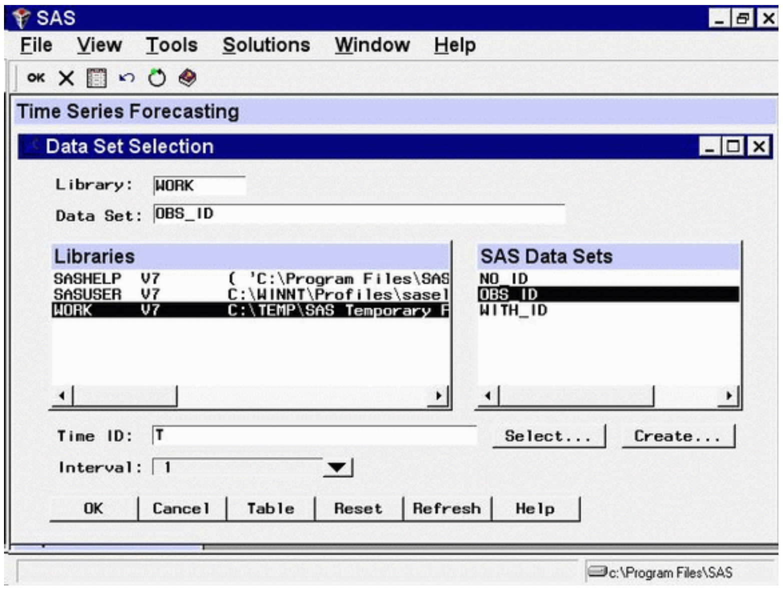
Time ID:

Interval:

Figure 35.5. Create Time ID Variable Window

Select the `OK` button. This brings up the `New Data Set Name` window. Enter "OBS_ID" in the `New data set name` field. Enter "T" in the `New ID variable name` field.

Now select the `OK` button. The new data set OBS_ID is created, and the system returns to the `Data Set Selection` window, which now appears as shown in Figure 35.6.



Library: WORK

Data Set: OBS_ID

```
SASHELP  V7     ( 'C:\Program Files\SAS
SASUSER  V7     C:\WINNT\Profiles\sasel
WORK     V7     C:\TEMP\SAS Temporary F
```

```
NO_ID
OBS_ID
WITH_ID
```

Time ID: T

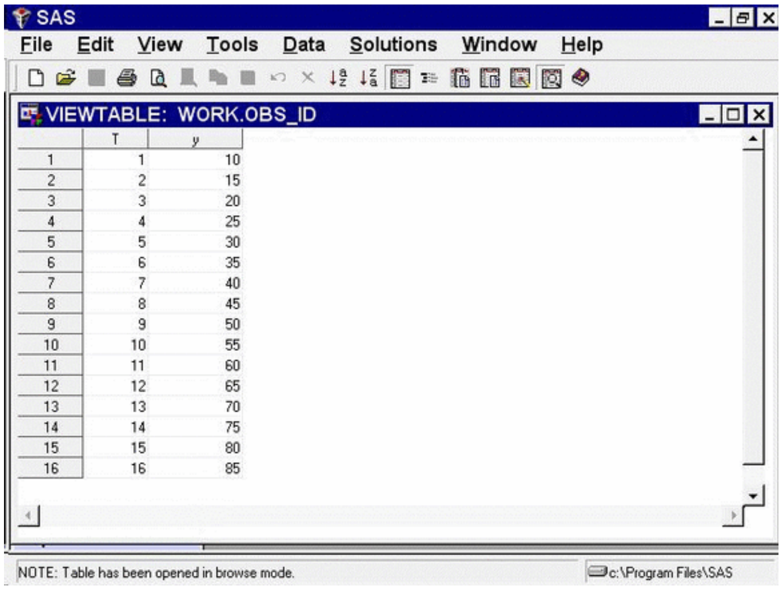
Interval: 1

c:\Program Files\SAS

Figure 35.6. Data Set Selection Window after Creating Time ID

The `Interval` field for OBS_ID has the value ‘1’. This means that the values of the time ID variable T increment by one between successive observations.

Select the `Table` button to look at the OBS_ID data set, as shown in Figure 35.7.



	T	y
1	1	10
2	2	15
3	3	20
4	4	25
5	5	30
6	6	35
7	7	40
8	8	45
9	9	50
10	10	55
11	11	60
12	12	65
13	13	70
14	14	75
15	15	80
16	16	85

NOTE: Table has been opened in browse mode.

c:\Program Files\SAS

Figure 35.7. VIEWTABLE of Data Set with Observation Index ID

Select `File` and `Close` to close the VIEWTABLE window. Select the `OK` button from the `Data Set Selection` window to return to the `Time Series Forecasting` window.

Creating a Time ID from Other Dating Variables

Your data set may contain ID variables that date the observations in a different way than the SAS date valued ID variable expected by the forecasting system. For example, for monthly data, the data set may contain the ID variables YEAR and MONTH, which together date the observations.

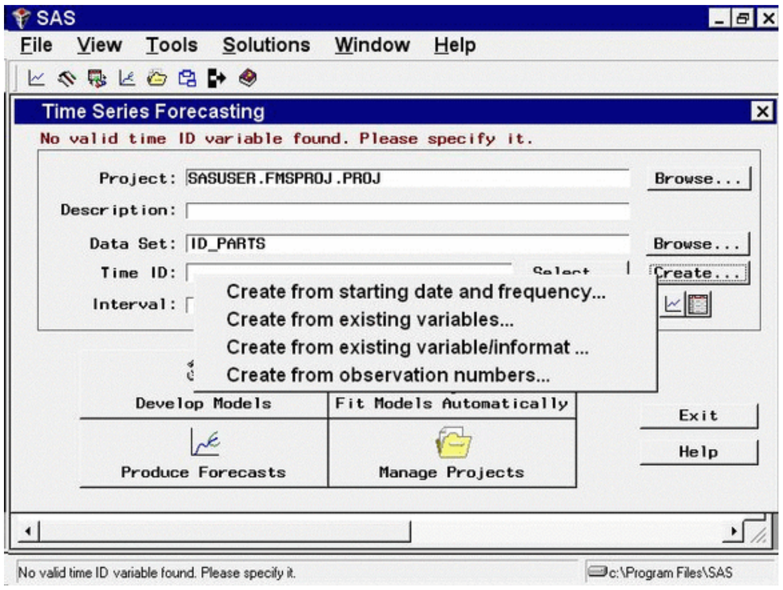
In these cases, you can use the Forecasting System's Create Time ID features to compute a time ID variable with SAS date values from the existing dating variables. As an example of this, you will use the SAS data set read in by the following SAS statements:

```
data id_parts;
  input yr qtr y;
datalines;
 91 1 10
 91 2 15
 91 3 20
 91 4 25
 92 1 30
 92 2 35
 92 3 40
 92 4 45
 93 1 50
 93 2 55
 93 3 60
 93 4 65
 94 1 70
 94 2 75
 94 3 80
 94 4 85
run;
```

Submit these SAS statements to create the data set ID_PARTS. This data set contains the three variables YR, QTR, and Y. YR and QTR are ID variables that together date the observations, but each variable provides only part of the date information. Because the forecasting system requires a single dating variable containing SAS date values, you need to combine YR and QTR to create a single variable DATE.

Type ID_PARTS in the `Data Set` field and press the ENTER key. (You could also use the Browse button to bring up the Data Set Selection window, as in the previous example, and complete this example from there.)

Select the Create button at the right of the `Time ID` field. This brings up the menu of Create Time ID choices, as shown in Figure 35.8.


SAS
File View Tools Solutions Window Help
Time Series Forecasting
Browse...
Browse...
Create...
Create from starting date and frequency...
Create from existing variables...
Create from existing variable/informat ...
Create from observation numbers...
Develop Models
Fit Models Automatically
Produce Forecasts
Manage Projects
Exit
Help
c:\Program Files\SAS

No valid time ID variable found. Please specify it.

Project: SASUSER.FMSPROJ.PROJ

Description:

Data Set: ID_PARTS

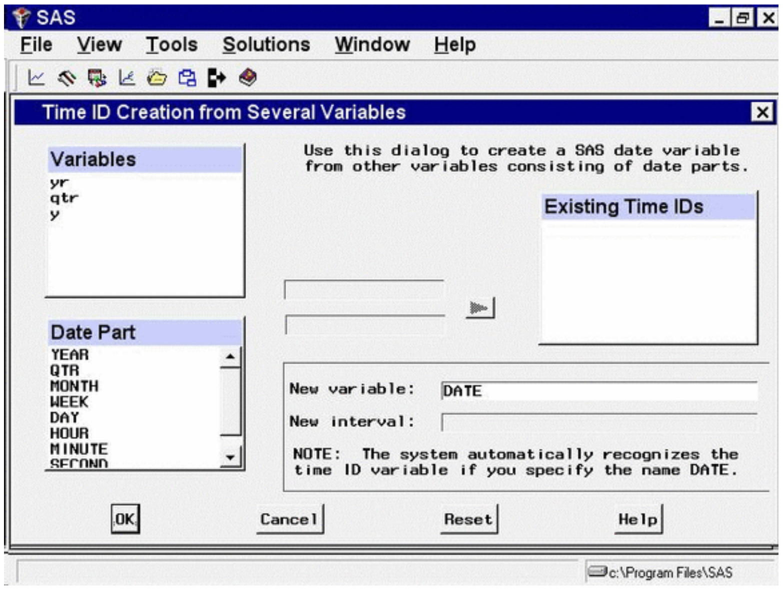
Time ID:

Interval:

No valid time ID variable found. Please specify it.

Figure 35.8. Adding a Time ID Variable

Select the second choice, `Create from existing variables`. This brings up the window shown in Figure 35.9.



Use this dialog to create a SAS date variable from other variables consisting of date parts.

New variable: DATE

New interval:

NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.

Figure 35.9. Creating a Time ID Variable from Date Parts

In the `Variables` list, select YR. In the `Date Part` list, select YEAR as shown in Figure 35.10.

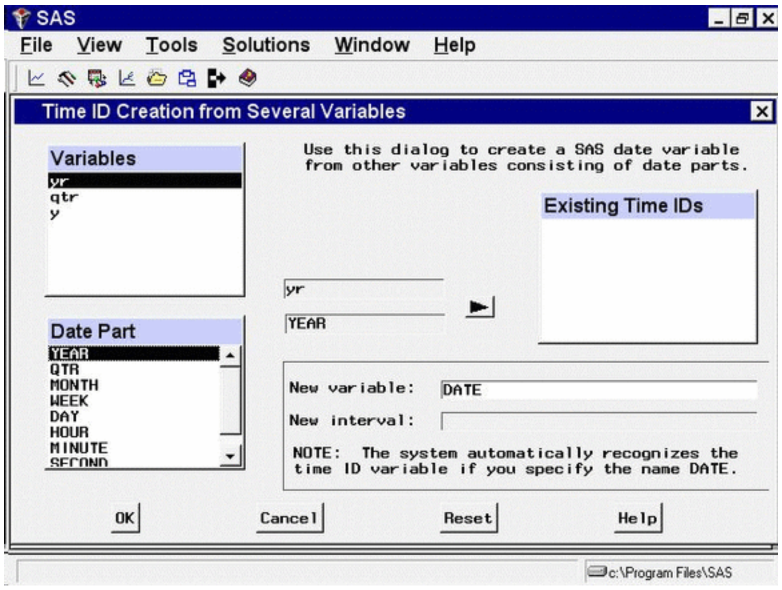

SAS
File View Tools Solutions Window Help
Time ID Creation from Several Variables
Variables
yr
qtr
y
Use this dialog to create a SAS date variable from other variables consisting of date parts.
Existing Time IDs
yr
YEAR
Date Part
YEAR
QTR
MONTH
WEEK
DAY
HOUR
MINUTE
SECOND
New variable: DATE
New interval:
NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.
OK
Cancel
Reset
Help
c:\Program Files\SAS

Figure 35.10. Specifying the ID Variable for Years

Now click the right-pointing arrow button. The variable YR and the part code YEAR are added to the `Existing Time IDs` list.

Next select QTR from the `Variables` list and select QTR from the `Date Part` list, and click the arrow button. This adds the variable QTR and the part code QTR to the `Existing Time IDs` list, as shown in Figure 35.11.

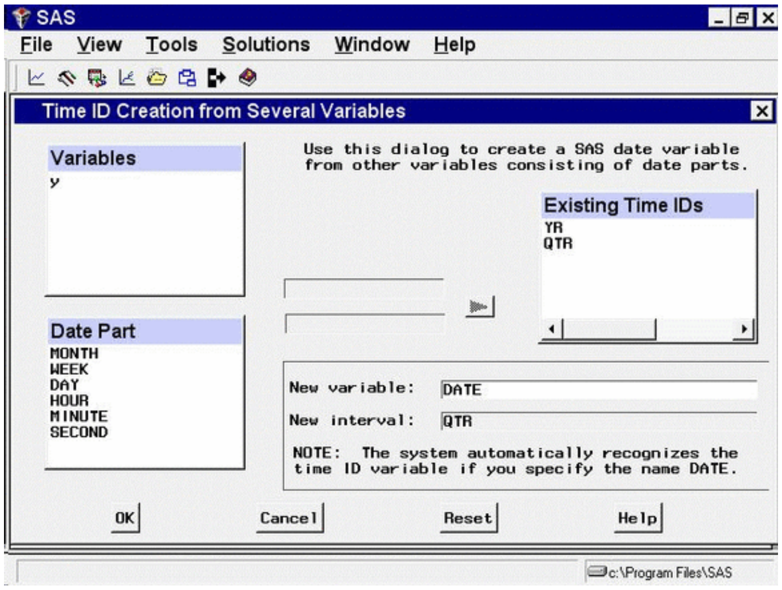

SAS
File View Tools Solutions Window Help
Time ID Creation from Several Variables
Variables
y
Use this dialog to create a SAS date variable from other variables consisting of date parts.
Existing Time IDs
YR
QTR
Date Part
MONTH
WEEK
DAY
HOUR
MINUTE
SECOND
New variable: DATE
New interval: QTR
NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.
OK
Cancel
Reset
Help
c:\Program Files\SAS

Figure 35.11. Creating a Time ID Variable from Date Parts

Now select the `OK` button. This brings up the `New Data Set Name` window. Change the `New data set name` field to NEWDATE, and then select the `OK` button.

The data set NEWDATE is created, and the system returns to the `Time Series Forecasting` window with NEWDATE as the selected Data Set. The Time ID field is set to DATE, and the Interval field is set to QTR.

Chapter 36
Specifying Forecasting Models

Chapter Contents

Chapter 36
Specifying Forecasting Models

This chapter explores the tools available through the Develop Models window for investigating the properties of time series and for specifying and fitting models. The first section shows you how to diagnose time series properties in order to determine the class of models appropriate for forecasting series with such properties. Later sections show you how to specify and fit different kinds of forecasting models.

Series Diagnostics

The series diagnostics tool helps you determine the kinds of forecasting models that are appropriate for the data series so that you can limit the search for the best forecasting model. The series diagnostics address these three questions: Is a log transformation needed to stabilize the variance? Is a time trend present in the data? Is there a seasonal pattern to the data?

The automatic model fitting process, which you used in the previous chapter through the Automatic Model Fitting window, performs series diagnostics and selects trial models from a list according to the results. You can also look at the diagnostic information and make your own decisions as to the kinds of models appropriate for the series. The following example illustrates the series diagnostics features.

Select "Develop Models" from the Time Series Forecasting window. Select the library SASHELP, the data set CITIMON, and the series RCARD. This series represents domestic retail sales of passenger cars. To look at this series, select "View Series" from the Develop Models window. This brings up the Time Series Viewer window, as shown in Figure 36.1.

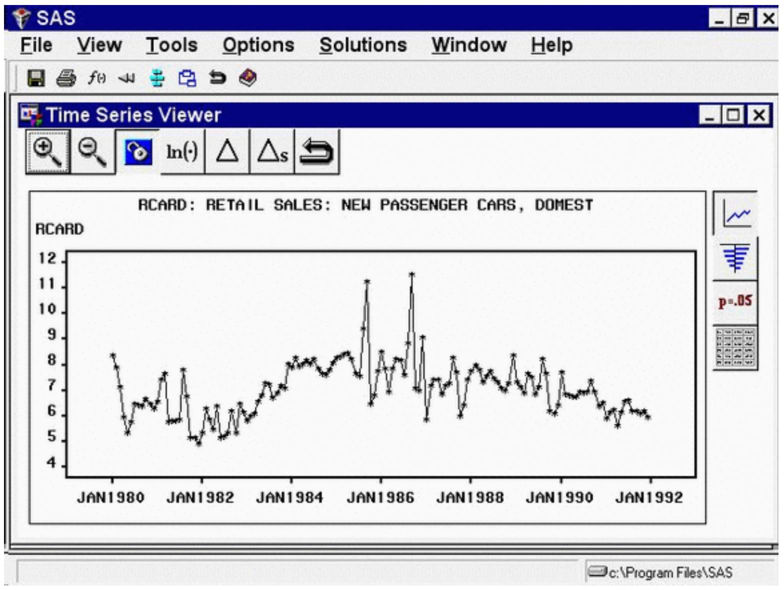

SAS
File View Tools Options Solutions Window Help
Time Series Viewer
ln(·) Δ Δs
RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
RCARD
12
11
10
9
8
7
6
5
4
JAN1980 JAN1982 JAN1984 JAN1986 JAN1988 JAN1990 JAN1992
p=.05
c:\Program Files\SAS

Figure 36.1. Automobile Sales Series

Select “Diagnose Series” from the Tools pull-down menu. You can do this from the Develop Models window or from the Time Series Viewer window. Figure 36.2 shows this from the Develop Models window.

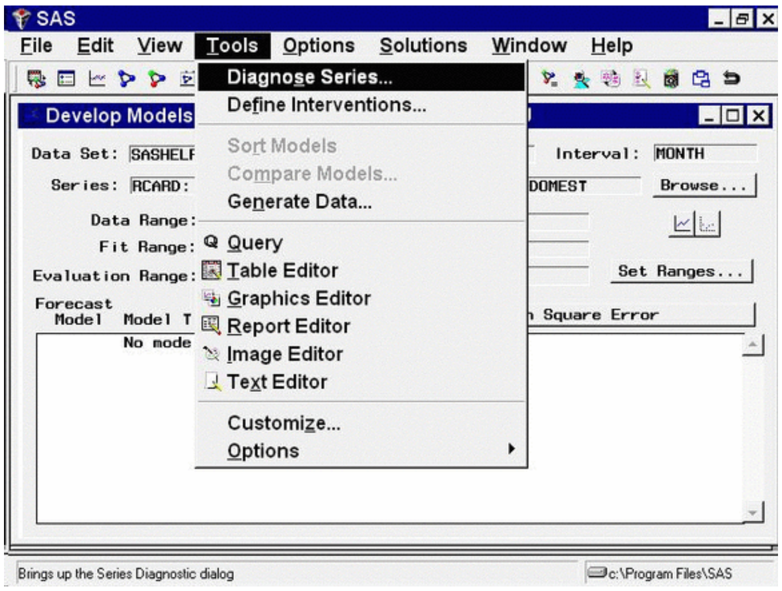
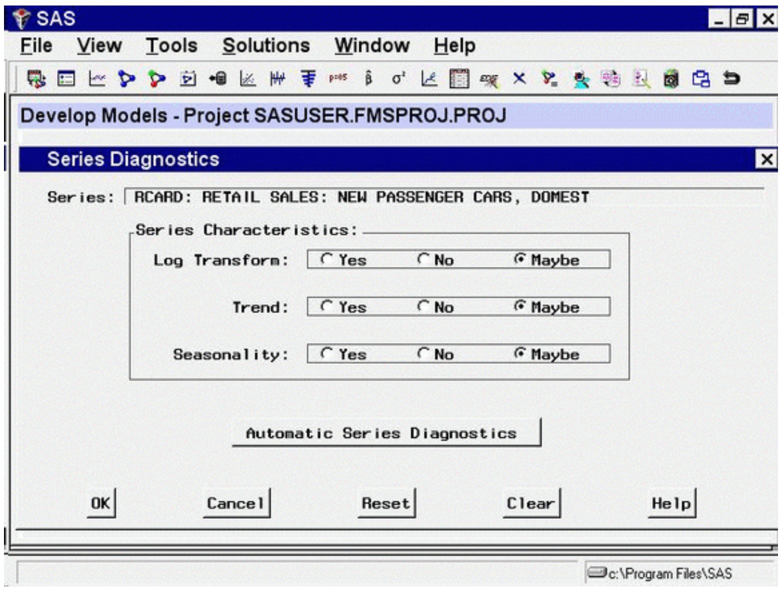


Figure 36.2. Selecting Series Diagnostics

This brings up the Series Diagnostics window, as shown in Figure 36.3.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Series Diagnostics
Automatic Series Diagnostics
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Series Characteristics:

Log Transform: Yes No [✓ Maybe]

Trend: Yes No [✓ Maybe]

Seasonality: Yes No [✓ Maybe]

Figure 36.3. Series Diagnostics Window

Each of the three series characteristics–need for log transformation, presence of a trend, and seasonality–has a set of radio buttons for `Yes`, `No`, and `Maybe`. `Yes` indicates that the series has the characteristic and that forecasting models fit to the series should be able to model and predict this behavior. `No` indicates that you do not need to consider forecasting models designed to predict series with this characteristic.

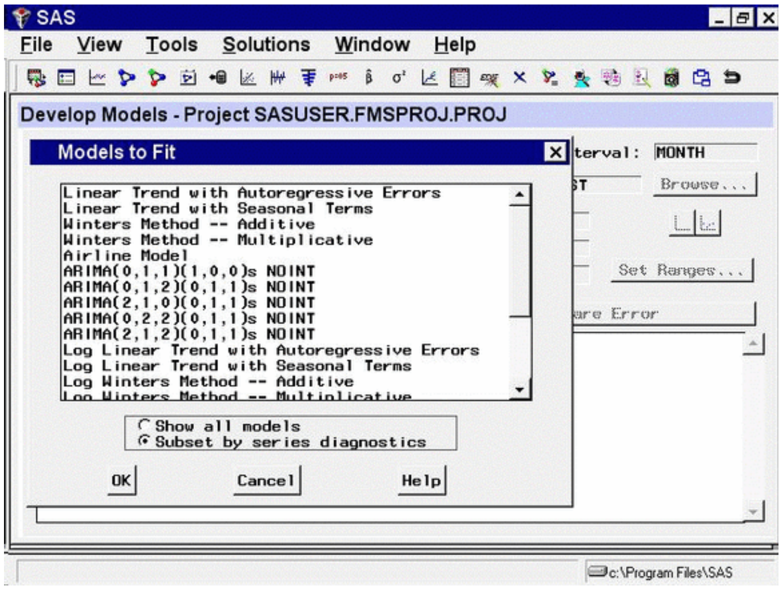
`Maybe` indicates that models with and without the characteristic should be considered. Initially, all these values are set to `Maybe.`

To have the system diagnose the series characteristics, select the Automatic Series Diagnostics button. This runs the diagnostic routines described in Chapter 41, "Forecasting Process Details," and sets the radio buttons according to the results. In this example, `Trend` and `Seasonality` are changed from `Maybe` to `Yes`, while `Log Transform` remains set to `Maybe`.

These diagnostic criteria affect the models displayed when you use the Models to Fit window or the Automatic Model Selection model-fitting options described in the following section. You can set the criteria manually, according to your judgment, by selecting any of the radio buttons, whether you have used the Automatic Series Diagnostics button or not. For this exercise, leave them as set by the automatic diagnostics. Select the OK button to close the Series Diagnostics window.

Models to Fit Window

As you saw in the previous chapter, you can select models from a list. Invoke the Models to Fit window by clicking the middle of the table and selecting "Fit Models from List" from the pop-up menu. This can also be selected from the tool bar or the Fit Model submenu of the Edit pull-down menu. The Models to Fit window comes up, as shown in Figure 36.4.



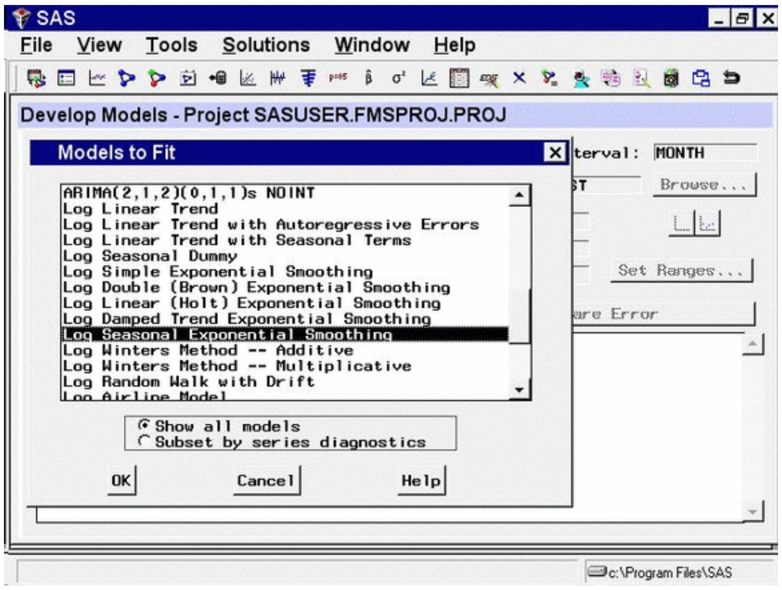
```
Linear Trend with Autoregressive Errors
Linear Trend with Seasonal Terms
Winters Method -- Additive
Winters Method -- Multiplicative
Airline Model
ARIMA(0,1,1)(1,0,0)s NOINT
ARIMA(0,1,2)(0,1,1)s NOINT
ARIMA(2,1,0)(0,1,1)s NOINT
ARIMA(0,2,2)(0,1,1)s NOINT
ARIMA(2,1,2)(0,1,1)s NOINT
Log Linear Trend with Autoregressive Errors
Log Linear Trend with Seasonal Terms
Log Winters Method -- Additive
Log Winters Method -- Multiplicative
```

Show all models [✓ Subset by series diagnostics]

Figure 36.4. Models to Fit Window

Since you have performed series diagnostics, the models shown are the subset that fits the diagnostic criteria.

Suppose you want to consider models other than those in this subset because you are undecided about including a trend in the model. Select the Show all Models radio button. Now the entire model selection list is shown. Scroll through the list until you find `Log Seasonal Exponential Smoothing`, as shown in Figure 36.5.

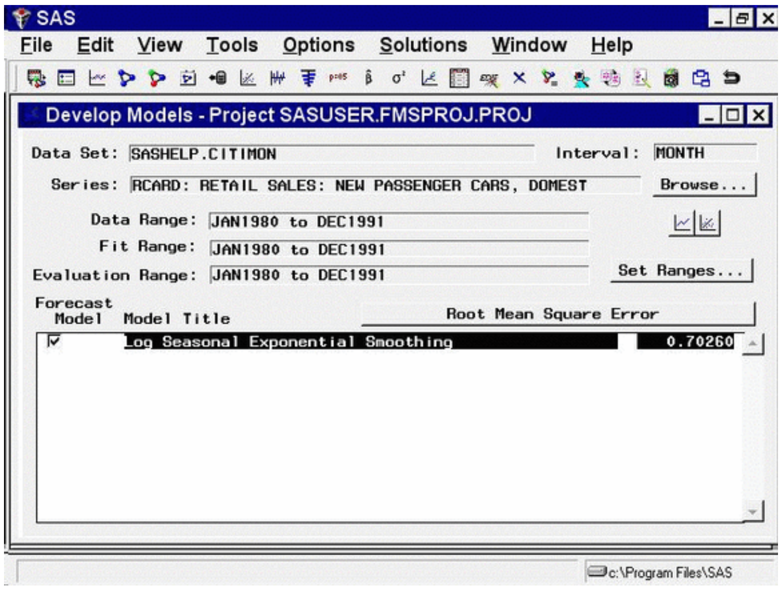


```
ARIMA(2,1,2)(0,1,1)s NOINT
Log Linear Trend
Log Linear Trend with Autoregressive Errors
Log Linear Trend with Seasonal Terms
Log Seasonal Dummy
Log Simple Exponential Smoothing
Log Double (Brown) Exponential Smoothing
Log Linear (Holt) Exponential Smoothing
Log Damped Trend Exponential Smoothing
Log Seasonal Exponential Smoothing
Log Winters Method -- Additive
Log Winters Method -- Multiplicative
Log Random Walk with Drift
Log Airline Model
```

[✓ Show all models] Subset by series diagnostics

Figure 36.5. Selecting a Model from List

This is a nontrended model, which seems a good candidate. Select this model, and then select the OK button. The model is fit to the series and then appears in the table with the value of the selected fit criterion, as shown in Figure 36.6.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
c:\Program Files\SAS

Data Set: SASHELP.CITIMON Interval: MONTH
Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Data Range: JAN1980 to DEC1991
Fit Range: JAN1980 to DEC1991
Evaluation Range: JAN1980 to DEC1991

Forecast Model	Model Title	Root Mean Square Error
✓	Log Seasonal Exponential Smoothing	0.70260

Figure 36.6. Develop Models Window Showing Model Fit

You can edit the model list that appears in the Models to Fit window by selecting "Options" and "Model Selection List" from the menu bar or by selecting the Edit Model List tool bar icon. You can then delete models you are not interested in from the default list and add models using any of the model specification methods described in this chapter. When you save your project, the edited model selection list is saved in the project file. In this way, you can use the Select from List item and the Automatic Model Selection item to select models from a customized search set.

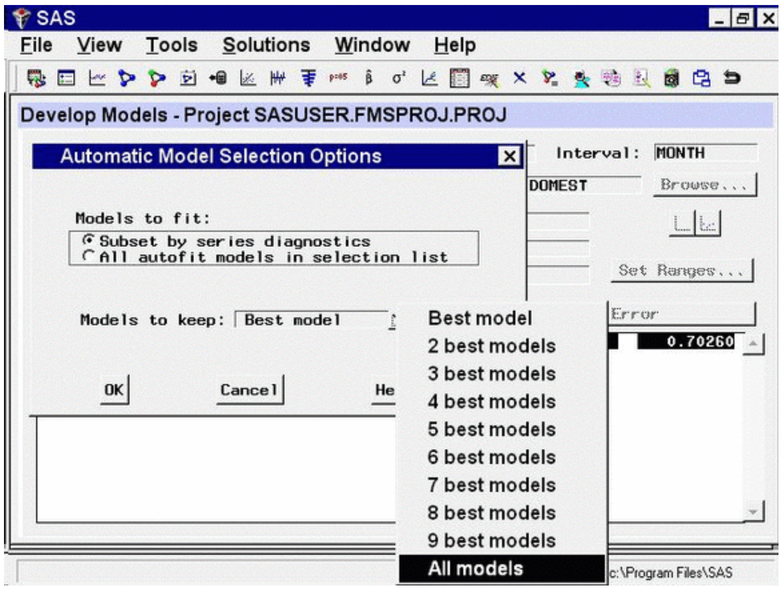
Automatic Model Selection

Automatic model selection is equivalent to choosing Select from List, as you did in the preceding section, fitting all the models in the subset list and then deleting all except the best fitting of the models. If series diagnostics have not yet been done, they are performed automatically to determine the model subset to fit. If you set the series diagnostics for log, trend, or seasonal criteria manually using the radio buttons, these choices are honored by the automatic fitting process.

Using automatic selection, the system does not pause to warn you of model fitting errors, such as failure of the estimates to converge (you can track these using the audit trail feature).

By default, only the best fitting model is kept. However, you can control the number of automatically fit models retained in the Develop Models list, and the following example shows how to do this.

From the menu bar, choose “Options” and “Automatic Fit.” This brings up the Automatic Model Selection Options window. Click the `Models to Keep` combo box arrow, and select “All models”, as shown in Figure 36.7. Now select “OK”.


Develop Models - Project SASUSER.FMSPROJ.PROJ

Automatic Model Selection Options

Models to fit:
- [✓ Subset by series diagnostics]
- All autofit models in selection list

Models to keep: Best model

Interval: MONTH

DOMEST

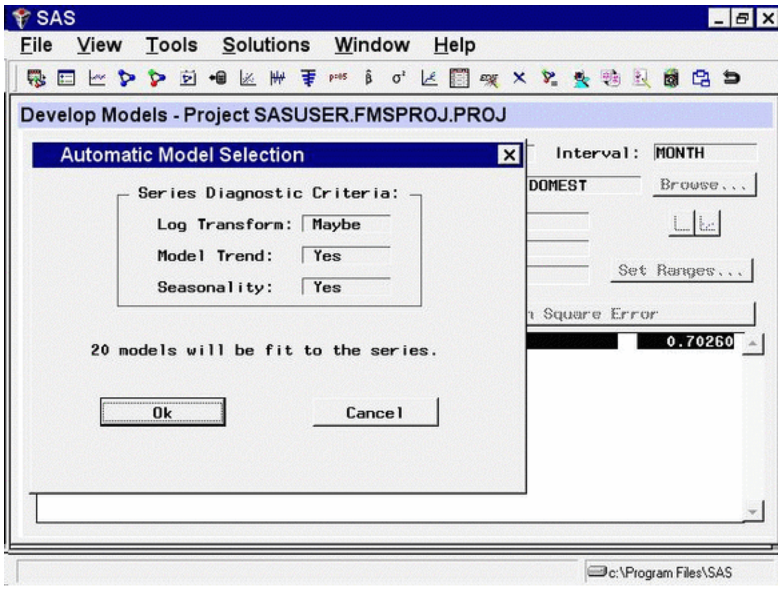
Error

0.70260

c:\Program Files\SAS

Figure 36.7. Selecting Number of Automatic Fit Models to Keep

Next, select “Fit Models Automatically” by clicking the middle of the table or using the toolbar or Edit pull-down menu. The Automatic Model Selection window appears, showing the diagnostic criteria in effect and the number of models to be fit, as shown in Figure 36.8.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
Square Error
Ok
Cancel
c:\Program Files\SAS

Automatic Model Selection

Interval: MONTH

DOMEST

Series Diagnostic Criteria:

Log Transform: Maybe

Model Trend: Yes

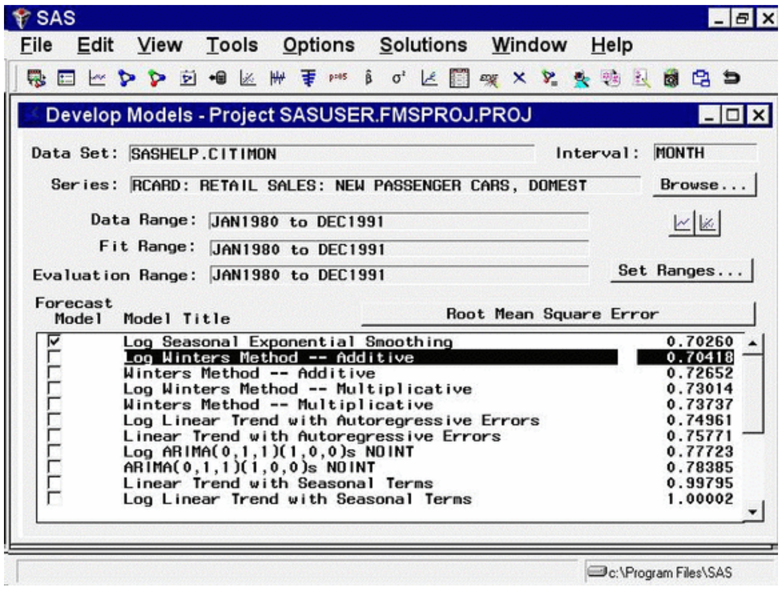
Seasonality: Yes

0.70260

20 models will be fit to the series.

Figure 36.8. Automatic Model Selection Window

Select the OK button. After the models have been fit, all of them appear in the table, in addition to the model which you fit earlier, as shown in Figure 36.9.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
c:\Program Files\SAS

Data Set: SASHELP.CITIMON

Interval: MONTH

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Data Range: JAN1980 to DEC1991

Fit Range: JAN1980 to DEC1991

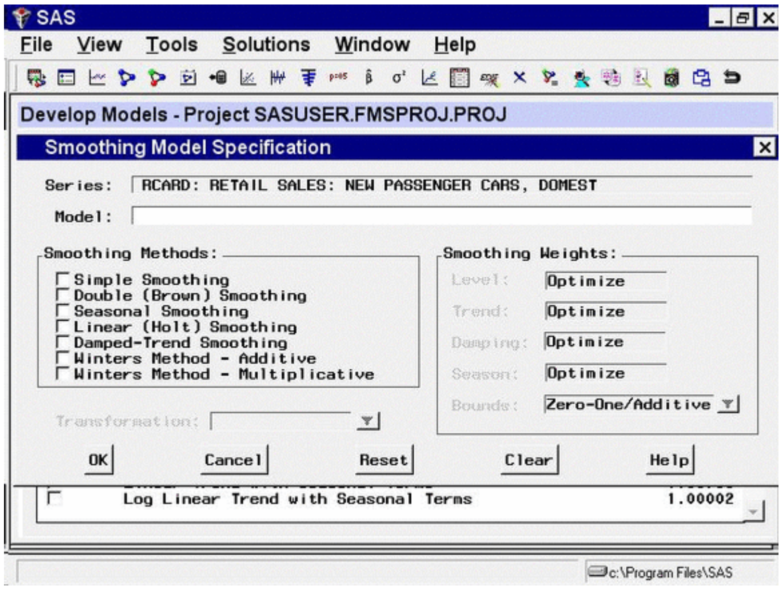
Evaluation Range: JAN1980 to DEC1991

Forecast Model	Model Title	Root Mean Square Error
✓	Log Seasonal Exponential Smoothing	0.70260
☐	Log Winters Method -- Additive	0.70418
☐	Winters Method -- Additive	0.72652
☐	Log Winters Method -- Multiplicative	0.73014
☐	Winters Method -- Multiplicative	0.73737
☐	Log Linear Trend with Autoregressive Errors	0.74961
☐	Linear Trend with Autoregressive Errors	0.75771
☐	Log ARIMA(0,1,1)(1,0,0)s NOINT	0.77723
☐	ARIMA(0,1,1)(1,0,0)s NOINT	0.78385
☐	Linear Trend with Seasonal Terms	0.99795
☐	Log Linear Trend with Seasonal Terms	1.00002

Figure 36.9. Automatically Fit Models

Smoothing Model Specification Window

To fit exponential smoothing and Winters models not already provided in the Models to Fit window, select “Fit Smoothing Model” from the pop-up menu or toolbar or select “Smoothing Model” from the Fit Model submenu of the Edit pull-down menu. This brings up the Smoothing Model Specification window, as shown in Figure 36.10.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Smoothing Model Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model:

Smoothing Methods:
- Simple Smoothing
- Double (Brown) Smoothing
- Seasonal Smoothing
- Linear (Holt) Smoothing
- Damped-Trend Smoothing
- Winters Method - Additive
- Winters Method - Multiplicative

Transformation:

Smoothing Weights:
Level: Optimize
Trend: Optimize
Damping: Optimize
Season: Optimize
Bounds: Zero-One/Additive

Log Linear Trend with Seasonal Terms 1.00002

Figure 36.10. Smoothing Model Specification Window

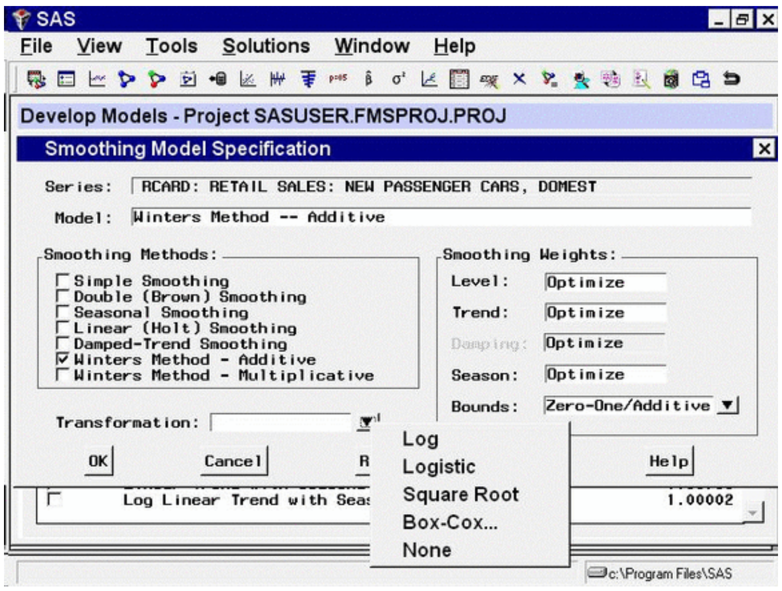
The Smoothing Model Specification window consists of several parts. At the top is the series name and a field for the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

The `Smoothing Methods` box lists the different methods available. Below the `Smoothing Methods` box is the Transformation field, which is used to apply the smoothing method to transformed series values.

The `Smoothing Weights` box specifies how the smoothing weights are determined. By default, the smoothing weights are automatically set to optimize the fit of the model to the data. See Chapter 41, “Forecasting Process Details,” for more information about how the smoothing weights are fit.

Under smoothing methods, select “Winters Method - Additive.” Notice the smoothing weights box to the right. The third item, `Damping`, is grayed out, while the other items, `Level`, `Trend`, and `Season`, show the word `Optimize`. This tells you that these three smoothing weights are applicable to the smoothing method that you selected and that the system is currently set to optimize these weights for you.

Next, specify a transformation using the `Transformation` combo box. A menu of transformation choices pops up, as shown in Figure 36.11.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Smoothing Model Specification
Log
Logistic
Square Root
Box-Cox...
None
OK
Cancel
Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model: Winters Method -- Additive

Smoothing Methods:
- [ ] Simple Smoothing
- [ ] Double (Brown) Smoothing
- [ ] Seasonal Smoothing
- [ ] Linear (Holt) Smoothing
- [ ] Damped-Trend Smoothing
- [✓] Winters Method - Additive
- [ ] Winters Method - Multiplicative

Smoothing Weights:

Level:	Optimize
Trend:	Optimize
Damping:	Optimize
Season:	Optimize
Bounds:	Zero-One/Additive

Transformation:

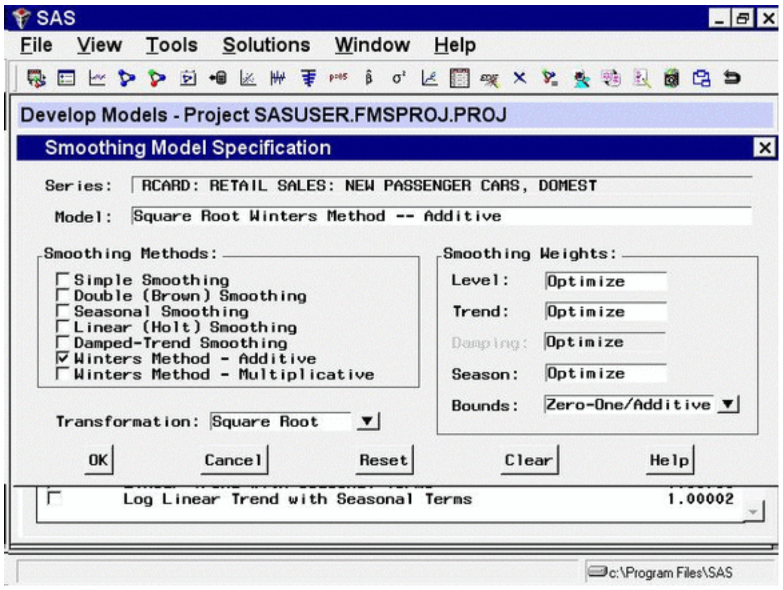
Log Linear Trend with Seas 1.00002

Figure 36.11. Transformation Options

You can specify a logarithmic, logistic, square root, or Box-Cox transformation. For this example, select “Square Root” from the pop-up menu. The Transformation field is now set to Square Root.

This means that the system will first take the square roots of the series values, apply the additive version of the Winters method to the square root series, and then produce the predictions for the original series by squaring the Winters method predictions (and multiplying by a variance factor if the Mean Prediction option is set in the Forecast Options window). See Chapter 41, “Forecasting Process Details,” for more information on predictions from transformed models.

The Smoothing Model Specification window should now appear as shown in Figure 36.12. Select the OK button to fit the model. The model is added to the table of fitted models in the Develop Models window.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Smoothing Model Specification
OK Cancel Reset Clear Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Model: Square Root Winters Method -- Additive

Smoothing Methods:

- [ ] Simple Smoothing
- [ ] Double (Brown) Smoothing
- [ ] Seasonal Smoothing
- [ ] Linear (Holt) Smoothing
- [ ] Damped-Trend Smoothing
- [✓] Winters Method - Additive
- [ ] Winters Method - Multiplicative

Smoothing Weights:

Level:	Optimize
Trend:	Optimize
Damping:	Optimize
Season:	Optimize
Bounds:	Zero-One/Additive

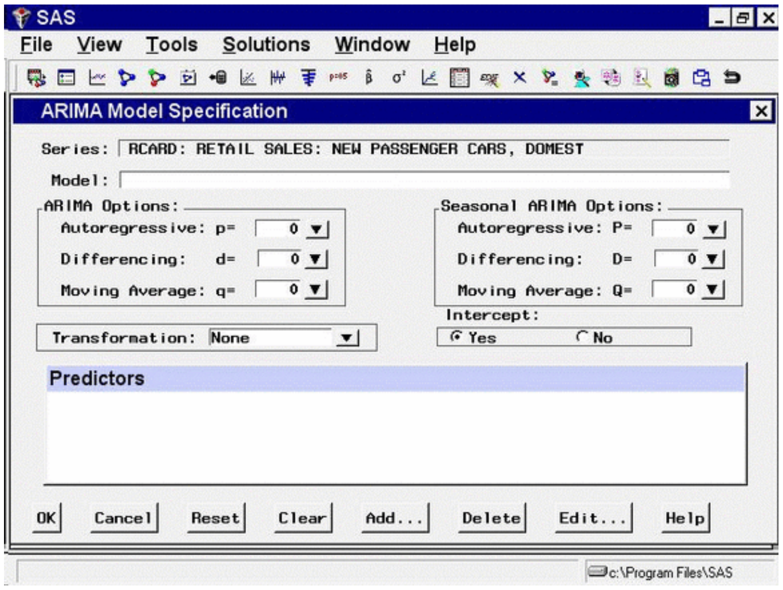
Transformation: Square Root

Log Linear Trend with Seasonal Terms 1.00002

Figure 36.12. Winter's Method applied to Square Root Series

ARIMA Model Specification Window

To fit ARIMA or Box-Jenkins models not already provided in the Models to Fit window, select the ARIMA model item from the pop-up menu, toolbar, or Edit pull-down menu. This brings up the ARIMA Model Specification window, as shown in Figure 36.13.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
Predictors
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Model:

ARIMA Options:

Autoregressive: p=	0
Differencing: d=	0
Moving Average: q=	0

Seasonal ARIMA Options:

Autoregressive: P=	0
Differencing: D=	0
Moving Average: Q=	0

Transformation: None

Intercept: [✓ Yes] No

Figure 36.13. ARIMA Model Specification Window

This ARIMA Model Specification window is structured according to the Box and Jenkins approach to time series modeling. You can specify the same time series models with the Custom Model Specification window and the ARIMA Model

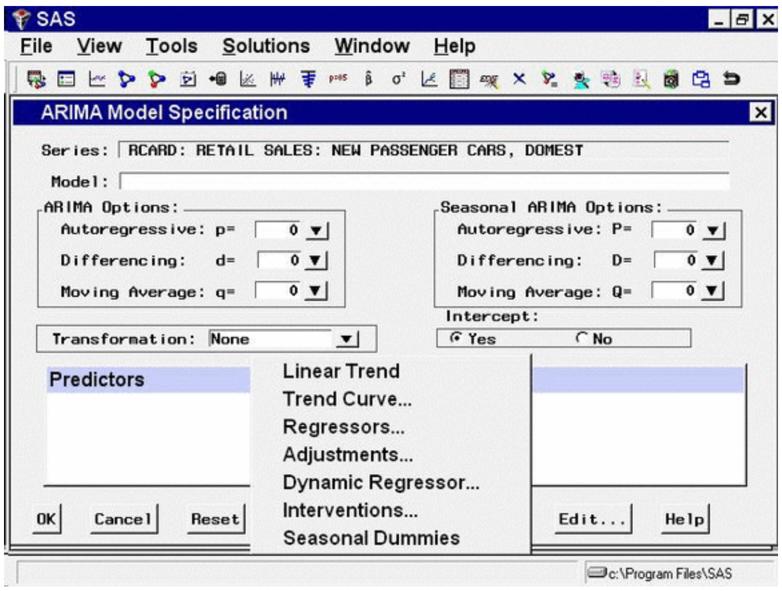
Specification window, but the windows are structured differently, and you may find one more convenient than the other.

At the top of the ARIMA Model Specification window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

Using the ARIMA Model Specification window, you can specify autoregressive (p), differencing (d), and moving average (q) orders for both simple and seasonal factors. You can specify transformations with the `Transformation` combo box. You can also specify whether an intercept is included in the ARIMA model.

In addition to specifying seasonal and nonseasonal ARIMA processes, you can also specify predictor variables and other terms as inputs to the model. ARIMA models with inputs are sometimes called ARIMAX models or Box-Tiao models. Another term for this kind of model is *dynamic regression*.

In the lower part of the ARIMA model specification window is the list of *predictors* to the model (initially empty). You can specify predictors using the Add button. This brings up a menu of different kinds of independent effects, as shown in Figure 36.14.



ARIMA Model Specification

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Model:

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Intercept: [✓ Yes] No

Transformation: None

Predictors

Figure 36.14. Add Predictors Menu

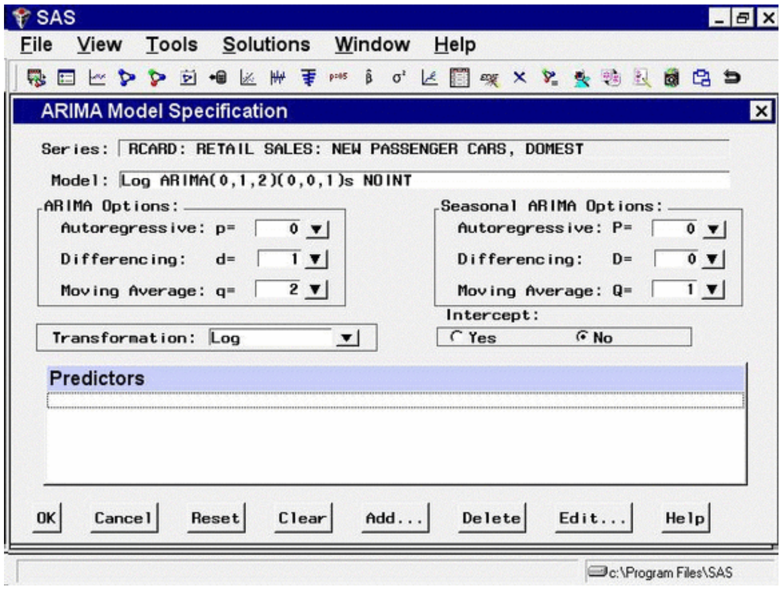
The kinds of predictor effects allowed include time trends, regressors, adjustments, dynamic regression (transfer functions), intervention effects, and seasonal dummy variables. How to use different kinds of predictors is explained in Chapter 38, "Using Predictor Variables."

As an example, in the `ARIMA Options` box, set the order of differencing `d` to `1` and the moving average order `q` to `2`. You can either type in these values or click the combo box arrows and select them from pop-up lists.

These selections specify an ARIMA(0,1,2) or IMA(1,2) model. (Refer to Chapter 11, "The ARIMA Procedure," for more information on the notation used for ARIMA models.) Notice that the model label at the top is now `IMA(1,2) NOINT`, meaning that the data are differenced once and a second-order moving average term is included with no intercept.

In the `Seasonal ARIMA Options` box, set the seasonal moving average order `Q` to `1`. This adds a first-order moving average term at the seasonal (12 month) lag. Finally, select "Log" in the Transformation combo box.

The model label is now `Log ARIMA(0,1,2)(0,0,1)s NOINT`, and the window appears as shown in Figure 36.15.



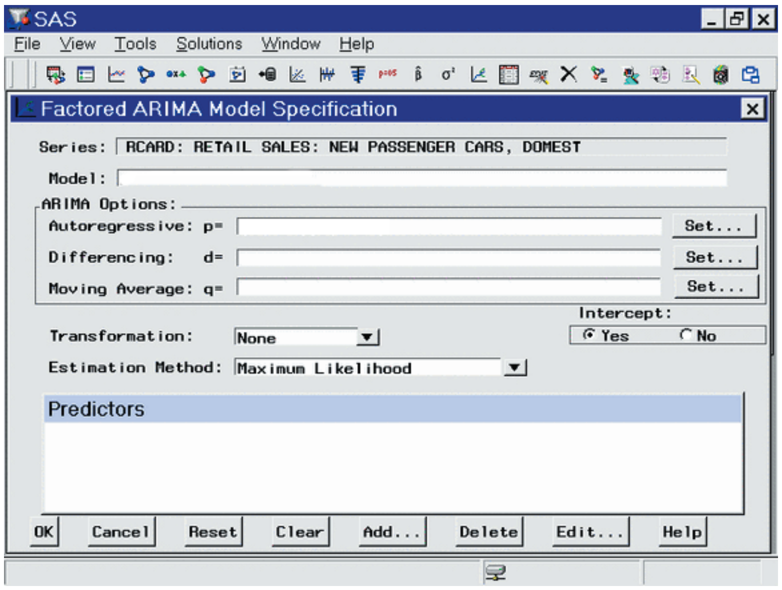
Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model: Log ARIMA(0,1,2)(0,0,1)s NOINT
ARIMA Options:
Autoregressive: p= 0
Differencing: d= 1
Moving Average: q= 2
Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 1
Intercept: Yes [✓ No]
Transformation: Log
Predictors

Figure 36.15. Log ARIMA(0,1,2)(0,0,1)s Specified

Select the OK button to fit the model. The model is fit and added to the Develop Models table.

Factored ARIMA Model Specification Window

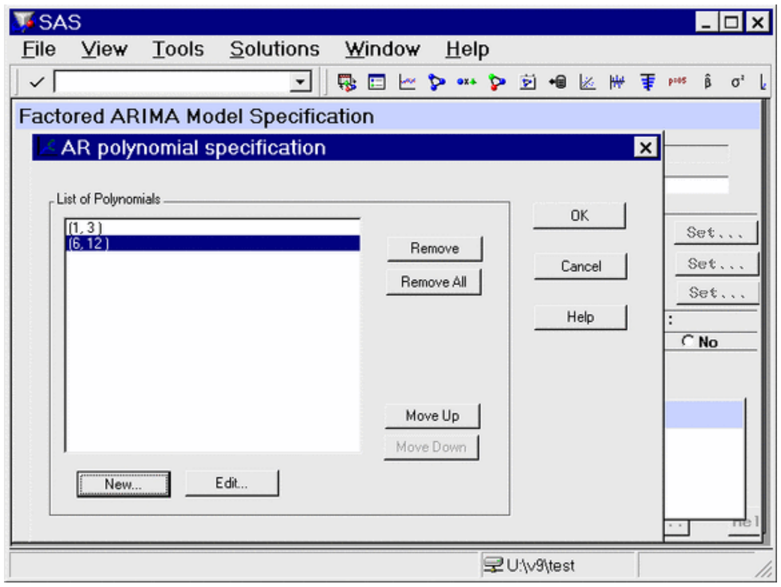
To fit a factored ARIMA model, select the Factored ARIMA model item from the pop-up menu, toolbar, or Edit pull-down menu. This brings up the Factored ARIMA Model Specification window, shown in Figure 36.16.



Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model:
ARIMA Options:
Autoregressive: p=
Differencing: d=
Moving Average: q=
Intercept: [✓ Yes] No
Transformation: None
Estimation Method: Maximum Likelihood

Figure 36.16. Factored ARIMA Model Specification Window

The Factored ARIMA Model Specification window is similar to the ARIMA Model Specification window, and has the same features, but uses a more general specification of the autoregressive (p), differencing (d), and moving average (q) terms. To specify these terms, select the corresponding Set button, as shown in Figure 36.16. For example, to specify autoregressive terms, select the first Set button. This brings up the AR Polynomial Specification Window, shown in Figure 36.17.


SAS
File View Tools Solutions Window Help
Factored ARIMA Model Specification
AR polynomial specification
Remove
Remove All
Move Up
Move Down
New...
Edit...
OK
Cancel
Help
Set...
Set...
Set...
No
U:\v9\test

List of Polynomials
(1, 3)
(6, 12)

Figure 36.17. AR Polynomial Specification Window

To add AR polynomial terms, select the New button. This brings up the Polynomial Specification Window, shown in Figure 36.18. Specify the first lag you want to include using the `Lag` spin box, then select the Add button. Repeat this process, adding each lag you want to include in the current list. All lags must be specified. For example, if you add only lag 3, the model contains only lag 3, not 1 through 3.

As an example, add lags 1 and 3, then select the OK button. The AR Polynomial Specification Window now shows (1,3) in the list of polynomials. Now select "New" again. Add lags 6 and 12 and select "OK". Now the AR Polynomial Specification Window shows (1,3) and (6,12) as shown in Figure 36.17. Select "OK" to close this window. The Factored ARIMA Model Specification Window now shows the factored model `p=(1,3)(6,12)`. Use the same technique to specify the q terms, or moving average part of the model. There is no limit to the number of lags or the number of factors you can include in the model.

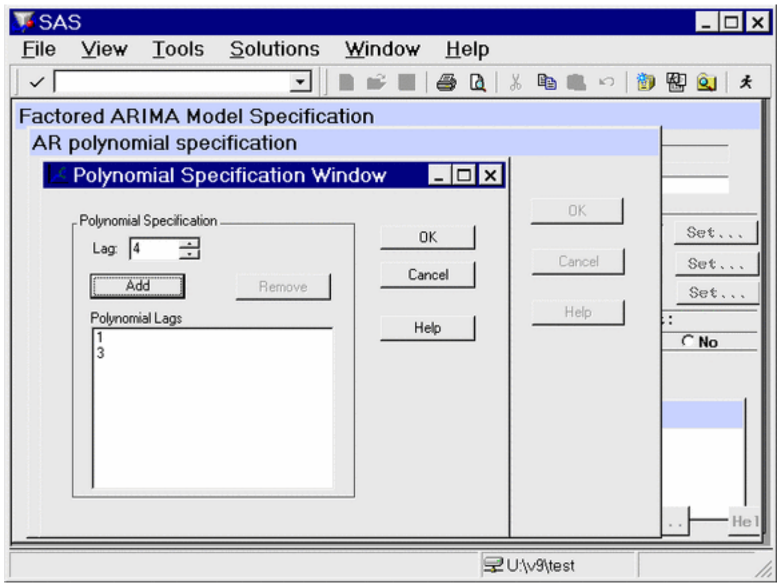

SAS
File View Tools Solutions Window Help
Factored ARIMA Model Specification
AR polynomial specification
Polynomial Specification Window
Polynomial Specification
Lag: 4
Add
Remove
Polynomial Lags
1
3
OK
Cancel
Help
Set...
Set...
Set...
No
U:\v9\test

Figure 36.18. Polynomial Specification Window

To specify differencing lags, select the middle Set button to bring up the Differencing Specification window. Specify lags using the spin box and add them to the list with the Add button. When you select "OK" to close the window, the differencing lags appear after `d=` in the Factored ARIMA Specification Window, within a single pair of parentheses.

You can use the Factored ARIMA Model Specification Window to specify any model that you can specify with the ARIMA Model and Custom Model windows, but the notation is more similar to that of the ARIMA Procedure (Chapter 11, "The ARIMA Procedure,"). Consider as an example the classic Airline model fit to the International Airline Travel series, `SASHELP.AIR`. This is a factored model with one moving average term at lag one and one moving average term at the seasonal lag, with first order differencing at the simple and seasonal lags. Using the ARIMA Model Specification Window, you specify the value 1 for the q and d terms and also for the Q and D terms, which represent the seasonal lags. For monthly data, the seasonal lags represent lag 12, since a yearly seasonal cycle is assumed.

By contrast, the Factored ARIMA Model Specification Window makes no assumptions about seasonal cycles. The Airline model is written as `IMA d=(1,12) q=(1)(12) NOINT`. To specify the differencing terms, add the values 1 and 12 in the Differencing Specification Window and select “OK”. Then select “New” in the MA Polynomial Specification Window, add the value 1, and select “OK”. To add the factored term, select “New” again, add the value 12, and select “OK”. Remember to select “No” in the Intercept radio box, since it is not selected by default. Select “OK” to close the Factored ARIMA Model Specification Window and fit the model.

You can show that the results are the same as they are when you specify the model using the ARIMA Model Specification Window and when you select Airline Model from the default model list. If you are familiar with the ARIMA Procedure (Chapter 11, “The ARIMA Procedure,”), you may wish to turn on the `Show Source Statements` option before fitting the model, then examine the procedure source statements in the log window after fitting the model.

The strength of the Factored ARIMA Specification approach lies in its ability to contruct unusual ARIMA models, such as:

Subset models
: These are models of order n, where fewer than n lags are specified. For example, an AR order 3 model might include lags 1 and 3 but not lag 2.

Unusual seasonal cycles
: For example, a monthly series might cycle two or four times per year instead of just once.

Multiple cycles
: For example, a daily sales series might peak on a certain day each week and also once a year at the Christmas season. Given sufficient data, you can fit a three-factor model, such as `IMA d=(1) q=(1)(7)(365)`.

Models with high order lags take longer to fit, and often fail to converge. To save time, select the Conditional Least Squares or Unconditional Least Squares estimation method (see Figure 36.16). Once you have narrowed down the list of candidate models, change to the Maximum Likelihood estimation method.

Custom Model Specification Window

To fit a custom time series model not already provided in the Models to Fit window, select the Custom Model item from the pop-up menu, toolbar, or Edit pull-down menu. This brings up the Custom Model Specification window, as shown in Figure 36.19.

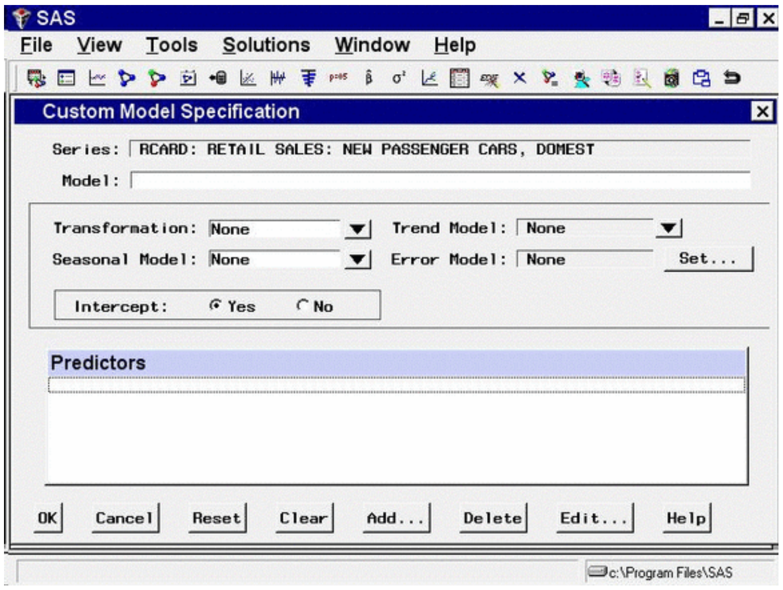

SAS
File View Tools Solutions Window Help
Custom Model Specification
Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model:
Transformation: None
Trend Model: None
Seasonal Model: None
Error Model: None
Intercept: [✓ Yes] No
Set...
Predictors
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Figure 36.19. Custom Model Specification Window

You can specify the same time series models with the Custom Model Specification window and the ARIMA Model Specification window, but the windows are structured differently, and you may find one more convenient than the other.

At the top of the Custom Model Specification window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

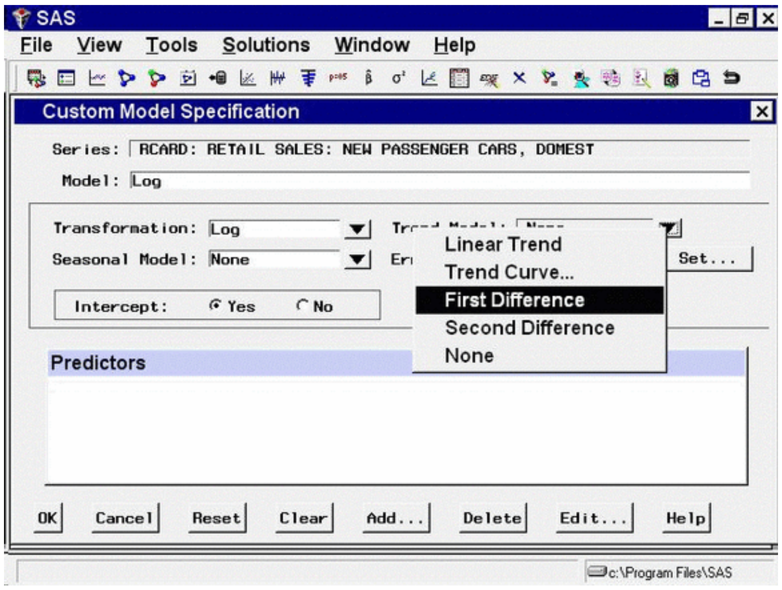
The middle part of the Custom Model Specification window consists of four fields: `Transformation`, `Trend Model`, `Seasonal Model`, and `Error Model`. These fields allow you to specify the model in four parts. Each part specifies how a different aspect of the pattern of the time series is modeled and predicted.

The `Predictors` list at the bottom of the Custom Model Specification window allows you to include different kinds of predictor variables in the forecasting model. The Predictors feature for the Custom Model Specification window is like the Predictors feature for the ARIMA Model Specification window, except that time trend predictors are provided through the Trend Model field and seasonal dummy variable predictors are provided through the Seasonal Model field.

To illustrate how to use the Custom Model Specification window, the following example specifies the same model you fit using the ARIMA Model Specification window.

First, specify the data transformation to use. Select "Log" using the Transformation combo box.

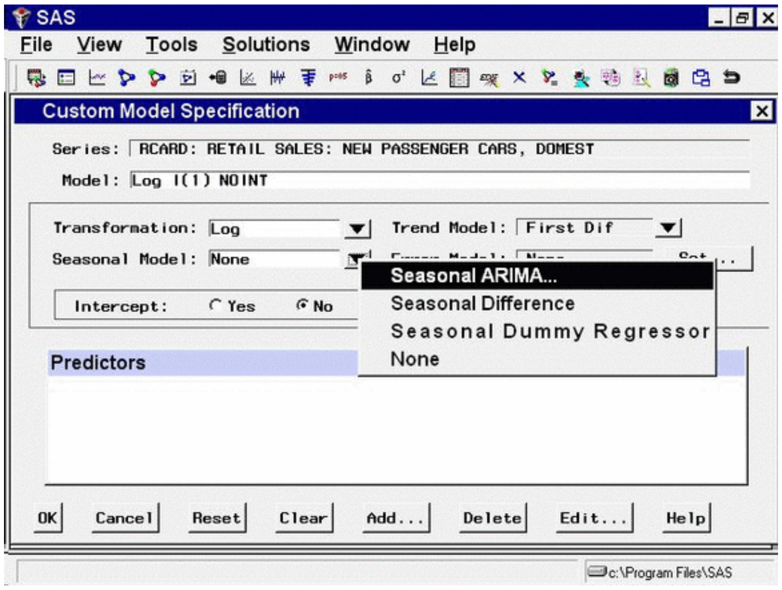
Second, specify how to model the trend in the series. Select `First Difference` in the `Trend Model` combo box, as shown in Figure 36.20.


SAS
File View Tools Solutions Window Help
Custom Model Specification
Linear Trend
Trend Curve...
First Difference
Second Difference
None
Predictors
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model: Log
Transformation: Log
Seasonal Model: None
Intercept: [✓ Yes] No

Figure 36.20. Trend Model Options

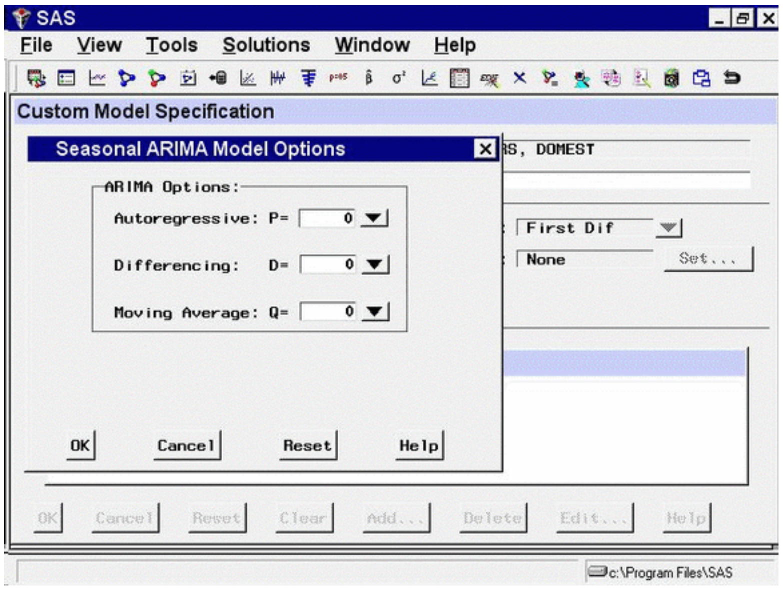
Next, specify how to model the seasonal pattern in the series. Select “Seasonal ARIMA” in the Seasonal Model combo box, as shown in Figure 36.21.


SAS
File View Tools Solutions Window Help
Custom Model Specification
Seasonal ARIMA...
Seasonal Difference
Seasonal Dummy Regressor
None
Predictors
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model: Log I(1) NOINT
Transformation: Log
Trend Model: First Dif
Seasonal Model: None
Intercept: Yes [✓ No]

Figure 36.21. Seasonal Model Options

This invokes the Seasonal ARIMA Model Options window, as shown in Figure 36.22.


SAS
File View Tools Solutions Window Help
Custom Model Specification
Seasonal ARIMA Model Options
OK Cancel Reset Help
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

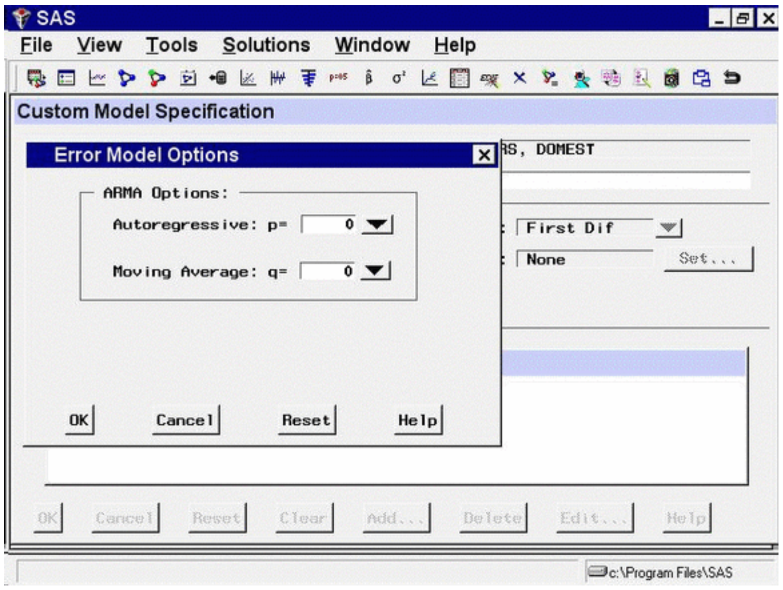
ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

RS, DOMEST
First Dif
None

Figure 36.22. Seasonal ARIMA Model Options

Specify a first-order seasonal moving average term by typing 1 or by selecting "1" from the Moving Average: Q= combo box pop-up menu, and then select the OK button.

Finally, specify how to model the autocorrelation pattern in the model prediction errors. Select the Set button to the right of the Error Model field. This invokes the Error Model Options window, as shown in Figure 36.23. This window allows you to specify an ARMA error process. Set the Moving Average order q to 2, and then select the OK button.

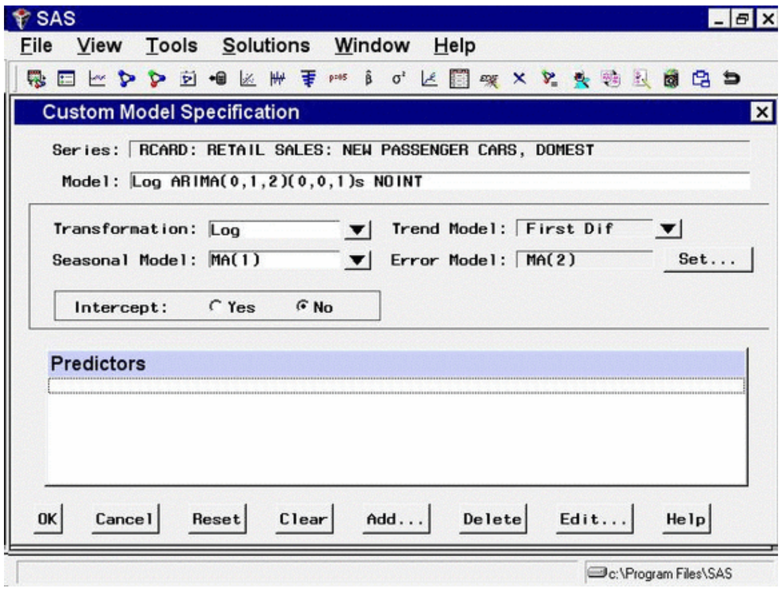

SAS
File View Tools Solutions Window Help
Custom Model Specification
Error Model Options
OK Cancel Reset Help
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

ARMA Options:
Autoregressive: p= 0
Moving Average: q= 0

RS, DOMEST
First Dif
None

Figure 36.23. Error Model Options

The Custom Model Specification window should now appear as shown in Figure 36.24. The model label at the top of the Custom Model Specification window should now read `Log ARIMA(0,1,2)(0,0,1)s` NOINT, just as it did when you used the ARIMA Model Specification window.


SAS
File View Tools Solutions Window Help
Custom Model Specification

Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST
Model: Log ARIMA(0,1,2)(0,0,1)s NOINT
Transformation: Log
Trend Model: First Dif
Seasonal Model: MA(1)
Error Model: MA(2)
Intercept: Yes [✓ No]

Figure 36.24. Log ARIMA(0,1,2)(0,0,1)s Specified

Now that you have seen how the Custom Model Specification window works, select "Cancel" to exit the window without fitting the model. This should return you to the Develop Models window.

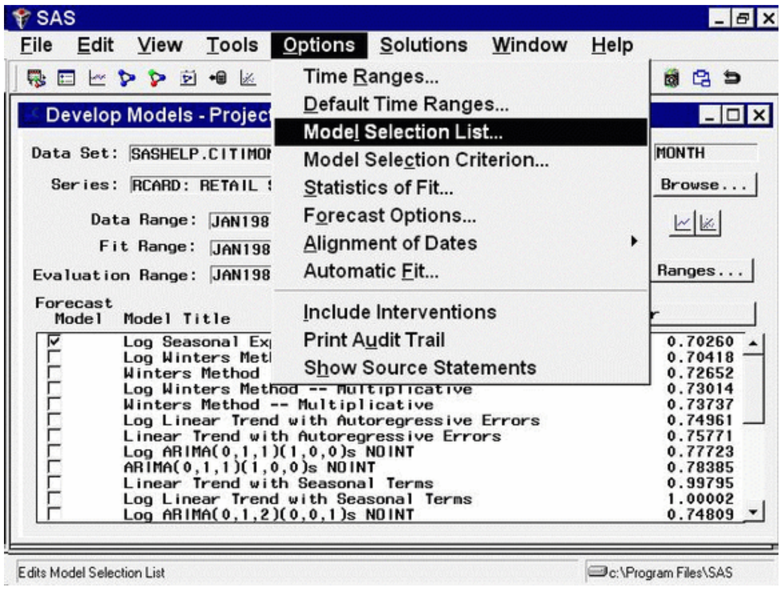
Editing the Model Selection List

Now that you know how to specify new models that are not included in the system default model selection list, you can edit the model selection list to add models that you expect to use in the future or to delete models that you do not expect to use. When you save the forecasting project to a SAS catalog, the edited model selection list is saved with the project file, and the list is restored when you load the project.

There are two reasons why you would add a model to the model selection list. First, by adding the model to the list, you will be able to fit the model to different time series by selecting it through the `Fit Models from List` action. You do not need to specify the model again every time you use it.

Second, once the model is added to the model selection list, it is available to the automatic model selection process. The model will then be considered automatically whenever you use the automatic model selection feature for any series.

To edit the model selection list, select "Model Selection List" from the Options pull-down menu as shown in Figure 36.25, or select the Edit Model List toolbar icon.

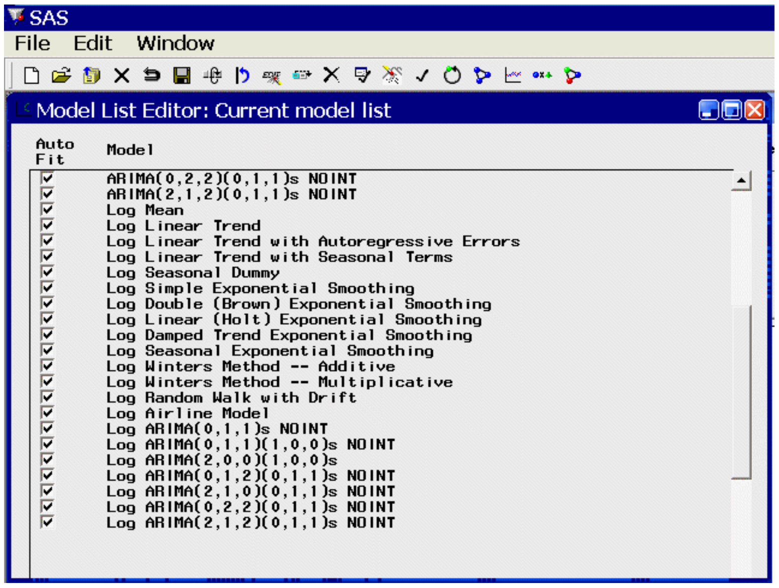


Data Set: SASHELP.CITIMON MONTH
Series: RCARD: RETAIL
Data Range: JAN198
Fit Range: JAN198
Evaluation Range: JAN198

```
Forecast
 Model   Model Title
 ☑       Log Seasonal Ex                                     0.70260
 ☐       Log Winters Met                                     0.70418
 ☐       Winters Method                                      0.72652
 ☐       Log Winters Method -- Multiplicative                0.73014
 ☐       Winters Method -- Multiplicative                    0.73737
 ☐       Log Linear Trend with Autoregressive Errors         0.74961
 ☐       Linear Trend with Autoregressive Errors             0.75771
 ☐       Log ARIMA(0,1,1)(1,0,0)s NOINT                      0.77723
 ☐       ARIMA(0,1,1)(1,0,0)s NOINT                          0.78385
 ☐       Linear Trend with Seasonal Terms                    0.99795
 ☐       Log Linear Trend with Seasonal Terms                1.00002
 ☐       Log ARIMA(0,1,2)(0,0,1)s NOINT                      0.74809
```

Figure 36.25. Model Selection List Option

This selection brings up the Model Selection List editor window, as shown in Figure 36.26. This window consists of the model selection list and an "Auto Fit" column, which controls for each model whether the model is included in the list of models used by the automatic model selection process.



```
Auto
Fit     Model
 ☑      ARIMA(0,2,2)(0,1,1)s NOINT
 ☑      ARIMA(2,1,2)(0,1,1)s NOINT
 ☑      Log Mean
 ☑      Log Linear Trend
 ☑      Log Linear Trend with Autoregressive Errors
 ☑      Log Linear Trend with Seasonal Terms
 ☑      Log Seasonal Dummy
 ☑      Log Simple Exponential Smoothing
 ☑      Log Double (Brown) Exponential Smoothing
 ☑      Log Linear (Holt) Exponential Smoothing
 ☑      Log Damped Trend Exponential Smoothing
 ☑      Log Seasonal Exponential Smoothing
 ☑      Log Winters Method -- Additive
 ☑      Log Winters Method -- Multiplicative
 ☑      Log Random Walk with Drift
 ☑      Log Airline Model
 ☑      Log ARIMA(0,1,1)s NOINT
 ☑      Log ARIMA(0,1,1)(1,0,0)s NOINT
 ☑      Log ARIMA(2,0,0)(1,0,0)s
 ☑      Log ARIMA(0,1,2)(0,1,1)s NOINT
 ☑      Log ARIMA(2,1,0)(0,1,1)s NOINT
 ☑      Log ARIMA(0,2,2)(0,1,1)s NOINT
 ☑      Log ARIMA(2,1,2)(0,1,1)s NOINT
```

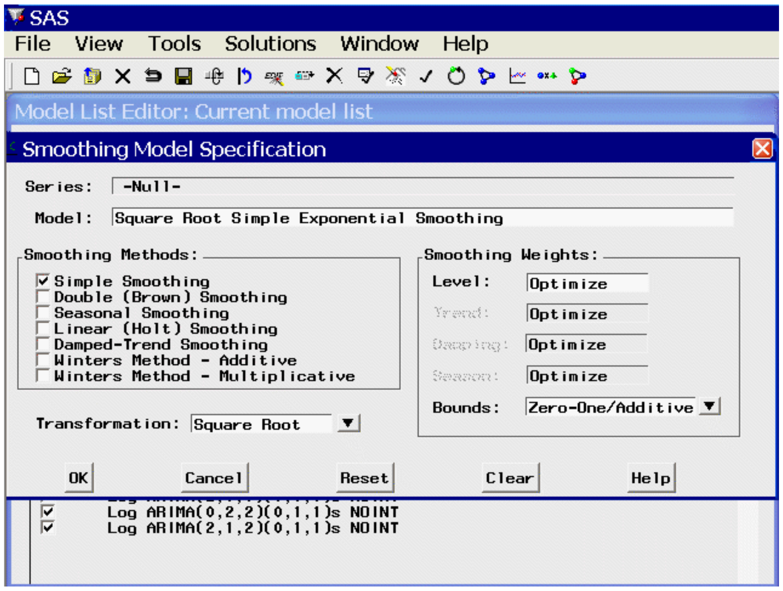
Figure 36.26. Model Selection List Window

To add a model to the list, select "Add Model" from the Edit pull-down menu and then select "Smoothing Model," "ARIMA Model," "Factored ARIMA Model," or "Custom Model" from the submenu. Alternatively, click the corresponding icon on the toolbar.

As an example, select "Smoothing Model." This brings up the Smoothing Model

Specification window. Note that the series name is “-Null-.” This means that you are not specifying a model to be fit to a particular series, but are specifying a model to be added to the selection list for later reference.

Specify a smoothing model. For example, select “Simple Smoothing” and then select the Square Root transformation. The window appears as shown in Figure 36.27.


SAS
File View Tools Solutions Window Help
Model List Editor: Current model list
Smoothing Model Specification
OK
Cancel
Reset
Clear
Help

Series: -Null-
Model: Square Root Simple Exponential Smoothing

Smoothing Methods:
- [✓] Simple Smoothing
- [ ] Double (Brown) Smoothing
- [ ] Seasonal Smoothing
- [ ] Linear (Holt) Smoothing
- [ ] Damped-Trend Smoothing
- [ ] Winters Method - Additive
- [ ] Winters Method - Multiplicative

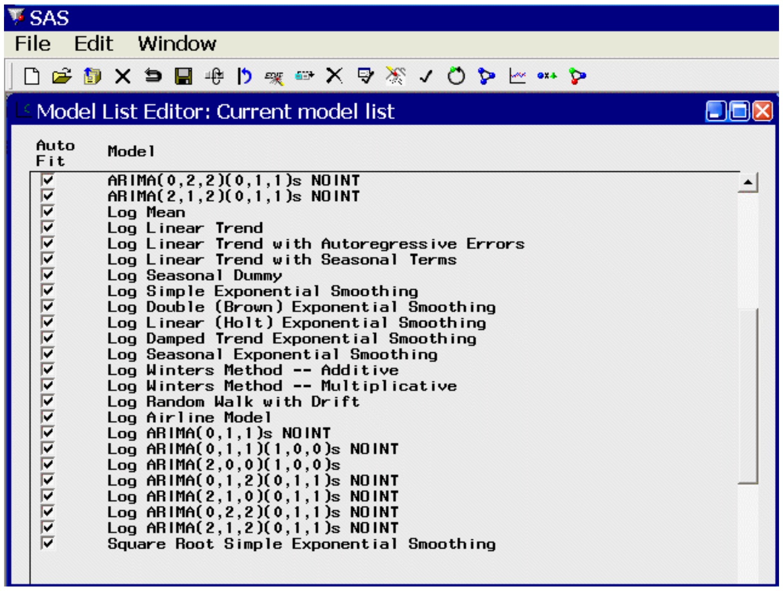
Transformation: Square Root

Smoothing Weights:
Level: Optimize
Trend: Optimize
Damping: Optimize
Season: Optimize
Bounds: Zero-One/Additive

```
✓   Log ARIMA(0,2,2)(0,1,1)s NOINT
✓   Log ARIMA(2,1,2)(0,1,1)s NOINT
```

Figure 36.27. Adding a Model Specification

Select the OK button to add the model to the end of the model selection list and return you to the Model Selection List window, as shown in Figure 36.28. You can now select the Fit Models from List model-fitting option to use the edited selection list.



```
Auto   Model
Fit
✓      ARIMA(0,2,2)(0,1,1)s NOINT
✓      ARIMA(2,1,2)(0,1,1)s NOINT
✓      Log Mean
✓      Log Linear Trend
✓      Log Linear Trend with Autoregressive Errors
✓      Log Linear Trend with Seasonal Terms
✓      Log Seasonal Dummy
✓      Log Simple Exponential Smoothing
✓      Log Double (Brown) Exponential Smoothing
✓      Log Linear (Holt) Exponential Smoothing
✓      Log Damped Trend Exponential Smoothing
✓      Log Seasonal Exponential Smoothing
✓      Log Winters Method -- Additive
✓      Log Winters Method -- Multiplicative
✓      Log Random Walk with Drift
✓      Log Airline Model
✓      Log ARIMA(0,1,1)s NOINT
✓      Log ARIMA(0,1,1)(1,0,0)s NOINT
✓      Log ARIMA(2,0,0)(1,0,0)s
✓      Log ARIMA(0,1,2)(0,1,1)s NOINT
✓      Log ARIMA(2,1,0)(0,1,1)s NOINT
✓      Log ARIMA(0,2,2)(0,1,1)s NOINT
✓      Log ARIMA(2,1,2)(0,1,1)s NOINT
✓      Square Root Simple Exponential Smoothing
```

Figure 36.28. Model Added to Selection List

If you want to delete one or more models from the list, select the model labels to highlight them in the list. Click a second time to unselect a selected model. Then select “Delete” from the Edit pull-down menu, or the corresponding toolbar icon. As an example, delete the `Square Root Simple Exponential Smoothing` model that you just added.

The Model Selection List editor window gives you a lot of flexibility for managing multiple model lists, as explained in the section “Model Selection List Editor Window” on page 2171. For example, you can create your own model lists from scratch or modify or combine previously saved model lists and those provided with the software, and you can save them and designate one as the default for future projects.

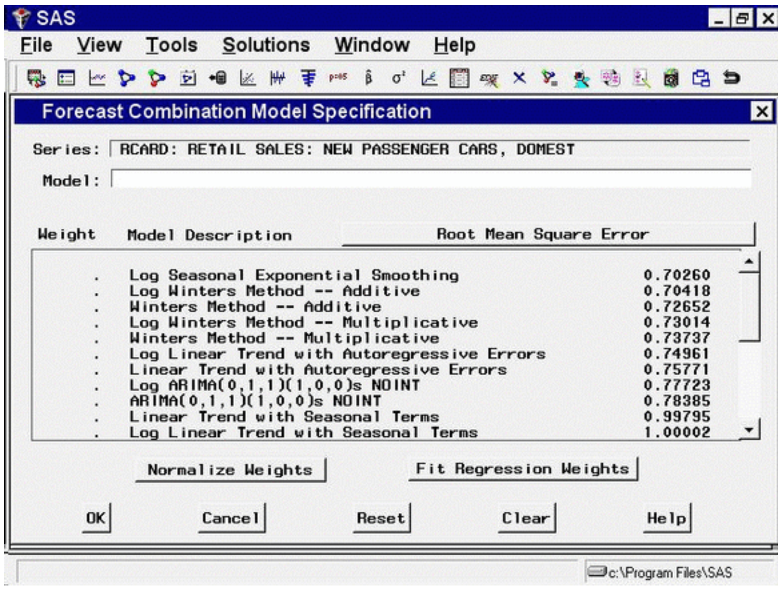
Now select “Close” from the File menu (or the Close icon) to close the Model Selection List editor window.

Forecast Combination Model Specification Window

Once you have fit several forecasting models to a series, you face the question of which model to use to produce the final forecasts. One possible answer is to combine or average the forecasts from several models. Combining the predictions from several different forecasting methods is a popular approach to forecasting.

The way that you produce forecast combinations with the Time Series Forecasting System is to use the Forecast Combination Model Specification window to specify a new forecasting model that performs the averaging of forecasts from the models you want to combine. This new model is added to the list of fitted models just like other models. You can then use the Model Viewer window features and Model Fit Comparison window features to examine the fit of the combined model.

To specify a forecast combination model, select “Combine Forecasts” from the pop-up menu or tool-bar, or select “Edit” amd “Fit Model” from the menu bar. This brings up the Forecast Combination Model Specification window, as shown in Figure 36.29.



Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Model:

Weight	Model Description	Root Mean Square Error
.	Log Seasonal Exponential Smoothing	0.70260
.	Log Winters Method -- Additive	0.70418
.	Winters Method -- Additive	0.72652
.	Log Winters Method -- Multiplicative	0.73014
.	Winters Method -- Multiplicative	0.73737
.	Log Linear Trend with Autoregressive Errors	0.74961
.	Linear Trend with Autoregressive Errors	0.75771
.	Log ARIMA(0,1,1)(1,0,0)s NOINT	0.77723
.	ARIMA(0,1,1)(1,0,0)s NOINT	0.78385
.	Linear Trend with Seasonal Terms	0.99795
.	Log Linear Trend with Seasonal Terms	1.00002

Figure 36.29. Forecast Combination Window

At the top of the Forecast Combination window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

The middle part of the Forecast Combination window consists of the list of models that you have fit to the series. This table shows the label and goodness-of-fit measure for each model and the combining weight assigned to the model.

The `Weight` column controls how much weight is given to each model in the combined forecasts. A missing weight means that the model is not used. Initially, all the models have missing Weight values.

You can enter the weight values you want to use in the Weight column. Alternatively, you can select models from the Model Description column, and Weight values for the models you select are set automatically. To remove a model from the combination, select it again. This resets its weight value to missing.

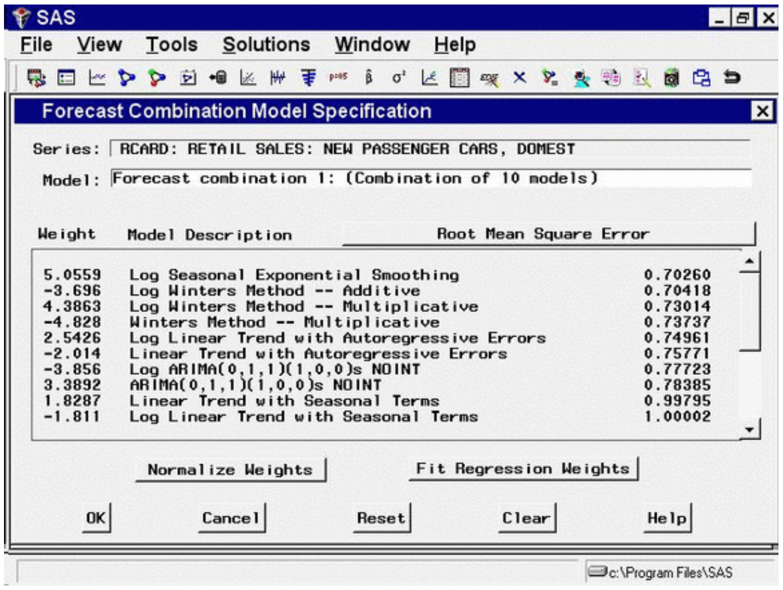
At the bottom of the Forecast Combination window are two buttons: `Normalize Weights` and `Fit Regression Weights.` The Normalize Weights button adjusts the nonmissing Weight values so that they sum to one. The Fit Regression Weights button uses linear regression to compute the weight values that produce the combination of model predictions with the best fit to the series.

If no models are selected, the Fit Regression Weights button fits weights for all the models in the list. You can compute regression weights for only some of the models by first selecting the models you want to combine and then selecting Fit Regression Weights. In this case, only the nonmissing Weight values are replaced with regression weights.

As an example of how to combine forecasting models, select all the models in the list. After you have finished selecting the models, all the models in the list should now have equal Weight values, which implies a simple average of the forecasts.

Now select the Fit Regression Weights button. The system performs a linear regression of the series on the predictions from the models with nonmissing Weight values and replaces the Weight values with the estimated regression coefficients. These are the combining weights that produce the smallest mean square prediction error within the sample.

The Forecast Combination window should now appear as shown in Figure 36.30. (Note that some of the regression weight values are negative.)



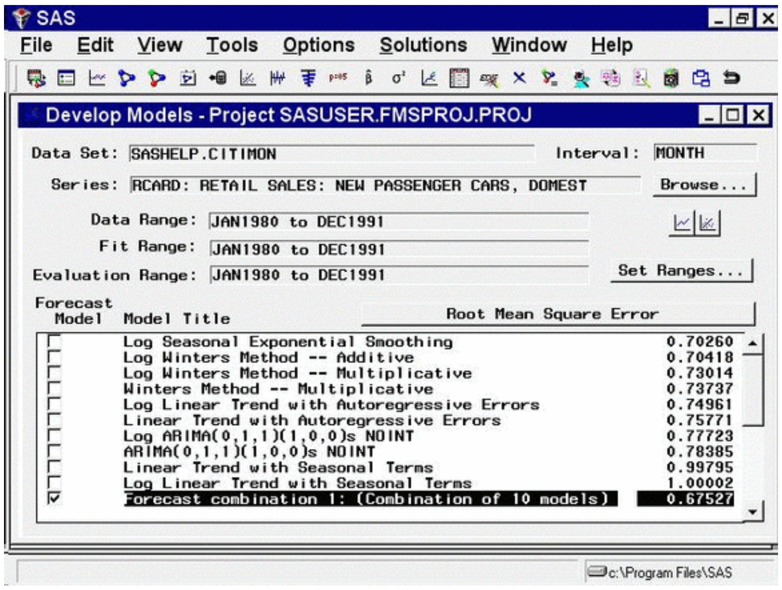
Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

Model: Forecast combination 1: (Combination of 10 models)

```
Weight   Model Description                            Root Mean Square Error

5.0559   Log Seasonal Exponential Smoothing                    0.70260
-3.696   Log Winters Method -- Additive                        0.70418
4.3863   Log Winters Method -- Multiplicative                  0.73014
-4.828   Winters Method -- Multiplicative                      0.73737
2.5426   Log Linear Trend with Autoregressive Errors           0.74961
-2.014   Linear Trend with Autoregressive Errors               0.75771
-3.856   Log ARIMA(0,1,1)(1,0,0)s NOINT                        0.77723
3.3892   ARIMA(0,1,1)(1,0,0)s NOINT                            0.78385
1.8287   Linear Trend with Seasonal Terms                      0.99795
-1.811   Log Linear Trend with Seasonal Terms                  1.00002
```

Figure 36.30. Combining Models

Select the OK button to fit the combined model. Now the Develop Models window shows this model to be the best fitting according to the root mean square error, as shown in Figure 36.31.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
c:\Program Files\SAS

```
Data Set: SASHELP.CITIMON                          Interval: MONTH
  Series: RCARD: RETAIL SALES: NEW PASSENGER CARS, DOMEST

      Data Range: JAN1980 to DEC1991
       Fit Range: JAN1980 to DEC1991
Evaluation Range: JAN1980 to DEC1991

Forecast
  Model   Model Title                                       Root Mean Square Error
  [ ]     Log Seasonal Exponential Smoothing                               0.70260
  [ ]     Log Winters Method -- Additive                                   0.70418
  [ ]     Log Winters Method -- Multiplicative                             0.73014
  [ ]     Winters Method -- Multiplicative                                 0.73737
  [ ]     Log Linear Trend with Autoregressive Errors                      0.74961
  [ ]     Linear Trend with Autoregressive Errors                          0.75771
  [ ]     Log ARIMA(0,1,1)(1,0,0)s NOINT                                   0.77723
  [ ]     ARIMA(0,1,1)(1,0,0)s NOINT                                       0.78385
  [ ]     Linear Trend with Seasonal Terms                                 0.99795
  [ ]     Log Linear Trend with Seasonal Terms                             1.00002
  [✓]     Forecast combination 1: (Combination of 10 models)               0.67527
```

Figure 36.31. Develop Models Window Showing All Models Fit

Notice that the combined model has a smaller root mean square error than any one of the models included in the combination. The confidence limits for forecast combinations are produced by taking a weighted average of the mean square prediction errors for the component forecasts, ignoring the covariance between the prediction errors.

Incorporating Forecasts from Other Sources

You may have forecasts from other sources that you want to include in the forecasting process. Examples of other forecasts you might want to use are "best guess" forecasts based on personal judgments, forecasts produced by government agencies or commercial forecasting services, planning scenarios, and reference or "base line" projections. Because such forecasts are produced externally to the Time Series Forecasting System, they are referred to as external forecasts.

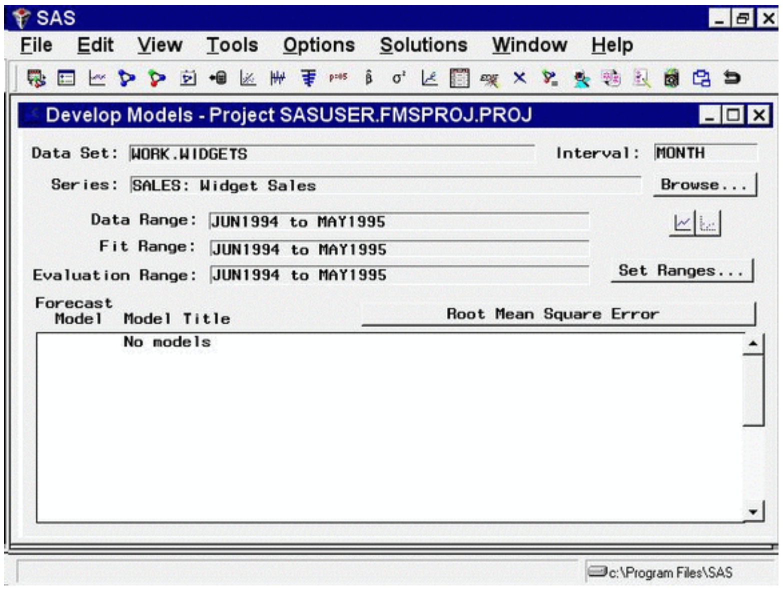
You can include external forecasts in combination models to produce compromise forecasts that split the difference between the external forecast and forecasting models that you fit. You can use external forecasts to compare them to the forecasts from models that are fit by the system.

To include external forecasts in the Time Series Forecasting process, you must first supply the external forecast as a variable in the input data set. You then specify a special kind of forecasting "model" whose predictions are identical to the external forecast recorded in the data set.

As an example, suppose you have 12 months of sales data and 5 months of sales forecasts based on a consensus opinion of the sales staff. The following statements create a SAS data set containing made-up numbers for this situation.

```
data widgets;
   input date monyy5. sales staff;
   format date monyy5.;
   label sales = "Widget Sales"
         staff = "Sales Staff Consensus Forecast";
   datalines;
jun94  142.1     .
jul94  139.6     .
aug94  145.0     .
sep94  150.2     .
oct94  151.1     .
nov94  154.3     .
dec94  158.7     .
jan95  155.9     .
feb95  159.2     .
mar95  160.8     .
apr95  162.0     .
may95  163.3     .
jun95      .  166.
jul95      .  168.
aug95      .  170.
sep95      .  171.
oct95      .  177.
run;
```

Submit the preceding statements in the SAS Program Editor window. From the Time Series Forecasting window, select "Develop Models". In the Series Selection window, select the data set WORK.WIDGETS and the variable SALES. The Develop Models window should now appear as shown in Figure 36.32.



Develop Models - Project SASUSER.FMSPROJ.PROJ

Data Set: WORK.WIDGETS Interval: MONTH
Series: SALES: Widget Sales
Data Range: JUN1994 to MAY1995
Fit Range: JUN1994 to MAY1995
Evaluation Range: JUN1994 to MAY1995

Forecast Model | Model Title | Root Mean Square Error
No models

Figure 36.32. Develop Models Window

Now select "Edit", "Fit Model", and "External Forecasts" from the menu bar of the Develop Models window, as shown in Figure 36.33, or the `Use External Forecasts` tool-bar icon.

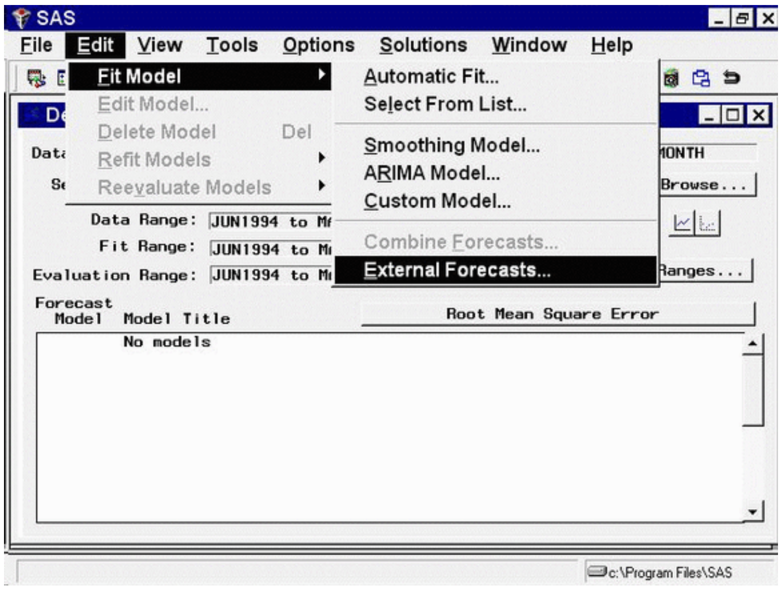
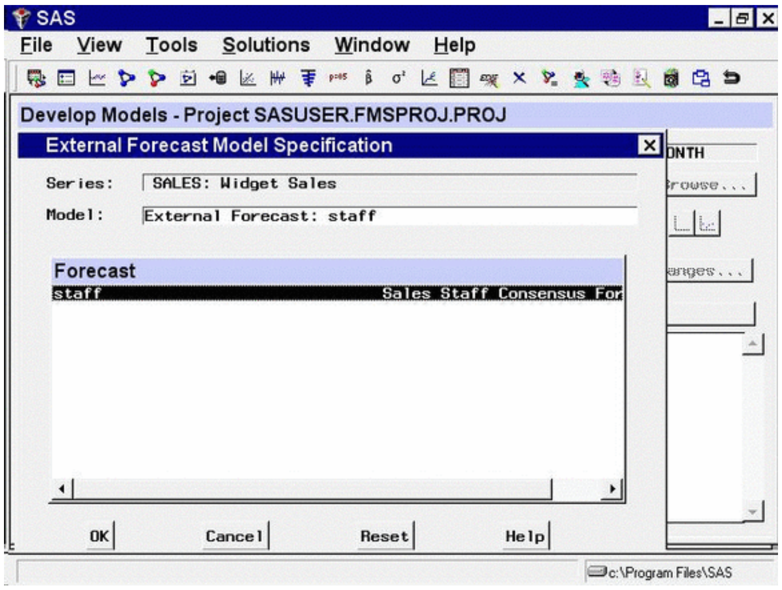


Figure 36.33. Adding a Model for an External Forecast Series

This selection brings up the External Forecast Model Specification window. Select the STAFF variable as shown in Figure 36.34.

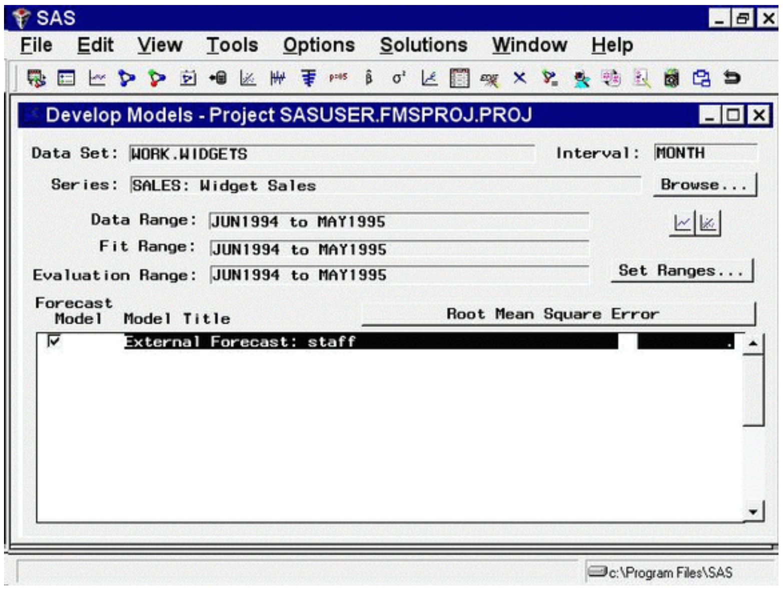

SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
External Forecast Model Specification
Forecast
OK
Cancel
Reset
Help
c:\Program Files\SAS

Series: SALES: Widget Sales
Model: External Forecast: staff

Forecast	
staff	Sales Staff Consensus For

Figure 36.34. External Forecast Series Selected

Select the OK button. The external forecast model is now "fit" and added to the Develop Models list, as shown in Figure 36.35.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ

Data Set: WORK.WIDGETS Interval: MONTH

Series: SALES: Widget Sales Browse...

Data Range: JUN1994 to MAY1995

Fit Range: JUN1994 to MAY1995

Evaluation Range: JUN1994 to MAY1995 Set Ranges...

Forecast Model	Model Title	Root Mean Square Error
✓	External Forecast: staff	.

Figure 36.35. Model for External Forecast

You can now use this model for comparison with the predictions from other forecasting models that you fit, or you can include it in a forecast combination model.

Note that no fitting is actually performed for an external forecast model. The predictions of the external forecast model are simply the values of the external forecast series read from the input data set. The goodness-of-fit statistics for such models will depend on the values that the external forecast series contains for observations within the period of fit. In this case, no STAFF values are given for past periods, and therefore the fit statistics for the model are missing.

Chapter 37
Choosing the Best Model

Chapter Contents

Chapter 37
Choosing the Best Forecasting Model

The Time Series Forecasting System provides a variety of tools for identifying potential forecasting models and for choosing the best fitting model. It allows you to decide how much control you want to have over the process, from a hands-on approach to one that is completely automated. This chapter begins with an exploration of the tools available through the Series Viewer and Model Viewer. It presents an example of identifying models graphically and exercising your knowledge of model properties. The remainder of the chapter shows you how to compare models by using a variety of statistics and by controlling the fit and evaluation time ranges. It concludes by showing you how to refit existing models and how to compare models using hold-out samples.

Time Series Viewer Features

The `Time Series Viewer` is a graphical tool for viewing and analyzing time series. It can be used separately from the `Time Series Forecasting System` using the TSVIEW command or by selecting `Time Series Viewer` from the `Analysis` pull-down menu under `Solutions`.

In this chapter you will use the Time Series Viewer to examine plots of your series before fitting models. Begin this example by invoking the forecasting system and selecting the `View Series Graphically` button, as shown in Figure 37.1, or the `View Series` toolbar icon.

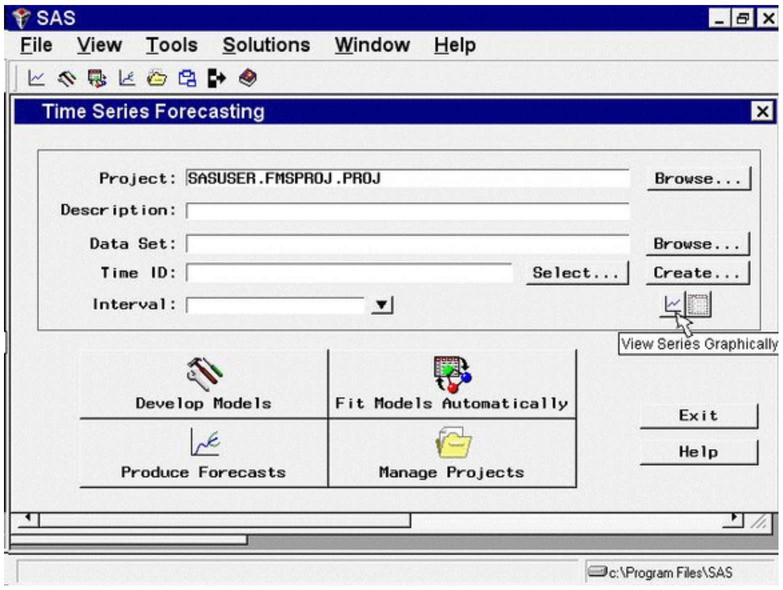


Figure 37.1. Invoking the Time Series Viewer

From the Series Selection window, select SASHELP as the library, WORKERS as the data set, and MASONRY as the time series, and then click the `Graph` button.

The Time Series Viewer displays a plot of the series, as shown in Figure 37.2.

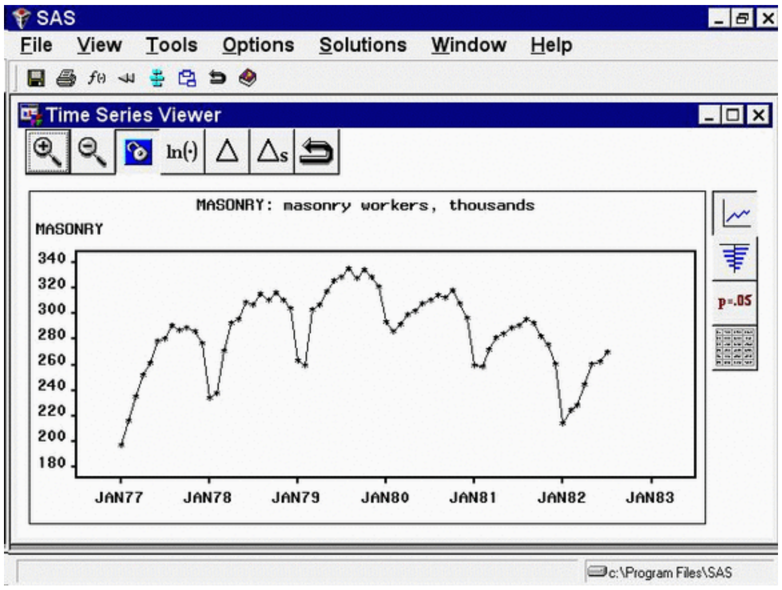

SAS
File View Tools Options Solutions Window Help
Time Series Viewer
ln(·) Δ Δs
MASONRY: masonry workers, thousands
MASONRY
340
320
300
280
260
240
220
200
180
JAN77 JAN78 JAN79 JAN80 JAN81 JAN82 JAN83
p=.05
c:\Program Files\SAS

Figure 37.2. Series Plot

Select the Zoom In icon, the first one on the window's horizontal toolbar. Notice that the mouse cursor changes shape and that "Note: Click on a corner of the region, then drag to the other corner" appears on the message line. Outline an area, as shown in Figure 37.3, by clicking the mouse at the upper-left corner, holding the button down, dragging to the lower right corner, and releasing the button.

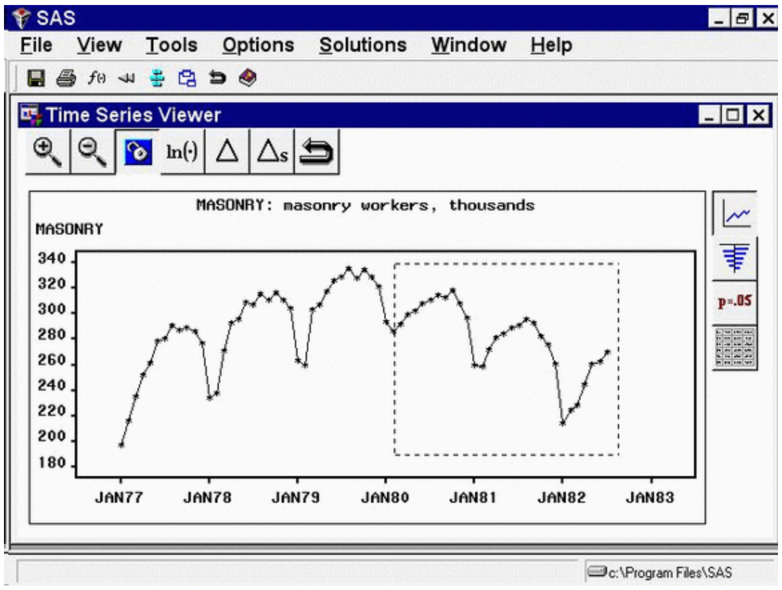

SAS
File View Tools Options Solutions Window Help
Time Series Viewer
ln(·) Δ Δs
MASONRY: masonry workers, thousands
MASONRY
340
320
300
280
260
240
220
200
180
JAN77 JAN78 JAN79 JAN80 JAN81 JAN82 JAN83
p=.05
c:\Program Files\SAS

Figure 37.3. Selecting an Area for Zoom

The zoomed plot should appear as shown in Figure 37.4.

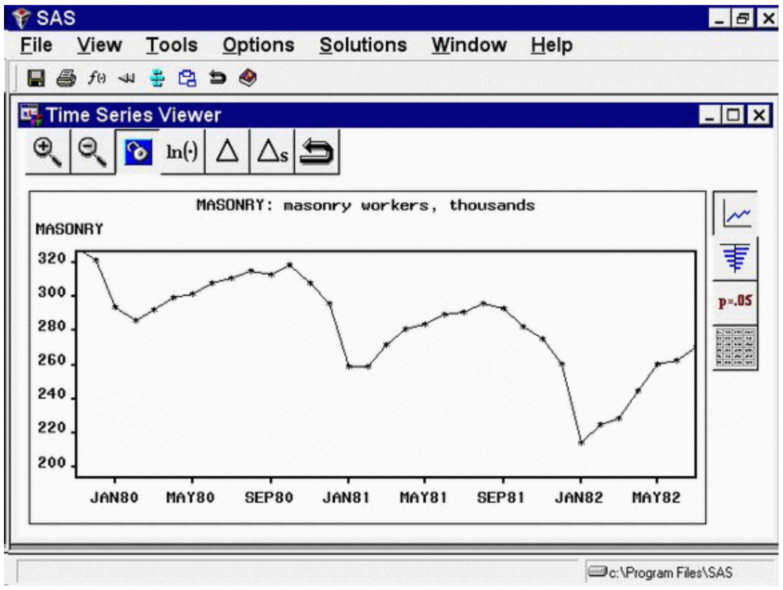


Figure 37.4. Zoomed Plot

You can repeat the process to zoom in still further. To return to the previous view, select the Zoom Out icon, the second icon on the window's horizontal toolbar.

The third icon on the horizontal toolbar is used to link or unlink the viewer window. By default, the viewer is linked, meaning that it is automatically updated to reflect selection of a different time series. To see this, return to the Series Selection window by clicking on it or using the Window pull-down menu or `Next Viewer` toolbar icon. Select the Electric series in the `Time Series Variables` list box. Notice that the Time Series Viewer window is updated to show a plot of the ELECTRIC series. Select the `Link/unlink` icon if you prefer to unlink the viewer so that it is not automatically updated in this way. Successive selections toggle between the linked and unlinked state. A note on the message line informs you of the state of the Time Series Viewer window.

When a Time Series Viewer window is linked, selecting `View Series` again will make the linked Viewer window active. When no Time Series Viewer window is linked, selecting `View Series` brings up an additional Time Series Viewer window. You can bring up as many Time Series Viewer windows as you want.

Having seen the plot in Figure 37.2, you might suspect that the series is nonstationary and seasonal. You can gain further insight into this by examining the sample autocorrelation function (ACF), partial autocorrelation function (PACF), and inverse autocorrelation function (IACF) plots. To switch the display to the autocorrelation plots, select the second icon from the top on the vertical toolbar at the right side of the Time Series Viewer. The plot appears as shown in Figure 37.5.

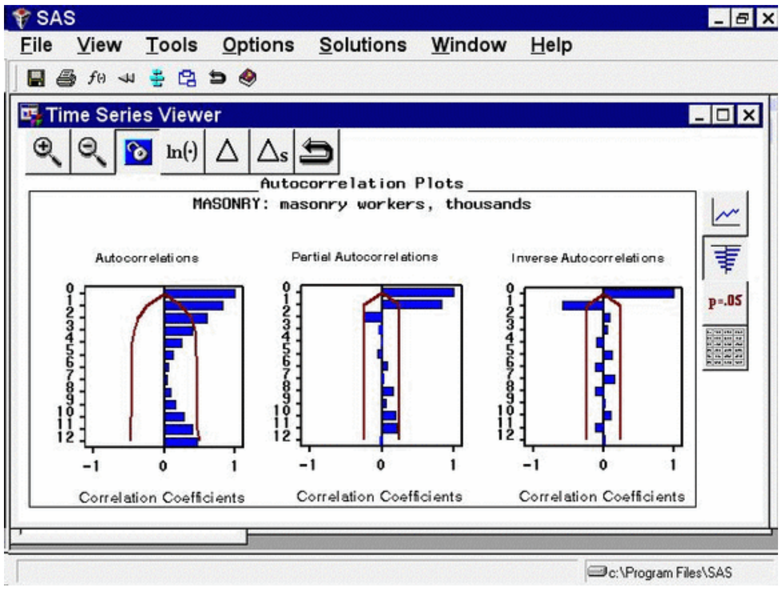


Figure 37.5. Sample Autocorrelation Plots

Each bar represents the value of the correlation coefficient at the given lag. The overlaid lines represent confidence limits computed at plus and minus two standard errors. You can switch the graphs to show significance probabilities by selecting `Correlation Probabilities` under the `Options` pull-down menu, or by selecting the `Toggle ACF Probabilities` toolbar icon.

The slow decline of the ACF suggests that first differencing may be warranted. To see the effect of first differencing, select the simple difference icon, the fifth icon from the left on the window's horizontal toolbar. The plot now appears as shown in Figure 37.6.

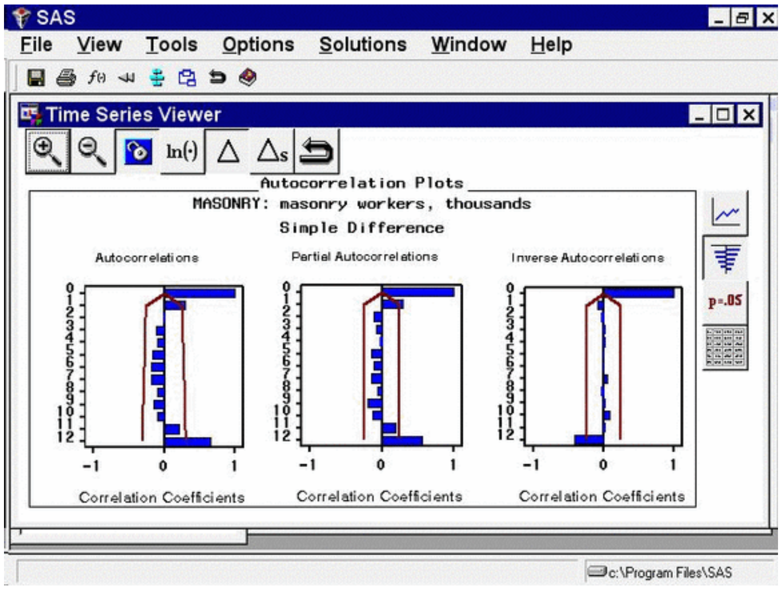


Figure 37.6. ACF Plots with First Difference Applied

Since the ACF still displays slow decline at seasonal lags, seasonal differencing is appropriate (in addition to the first differencing already applied). Select the `Seasonal Difference` icon, the sixth icon from the left on the horizontal toolbar. The plot now appears as shown in Figure 37.7.

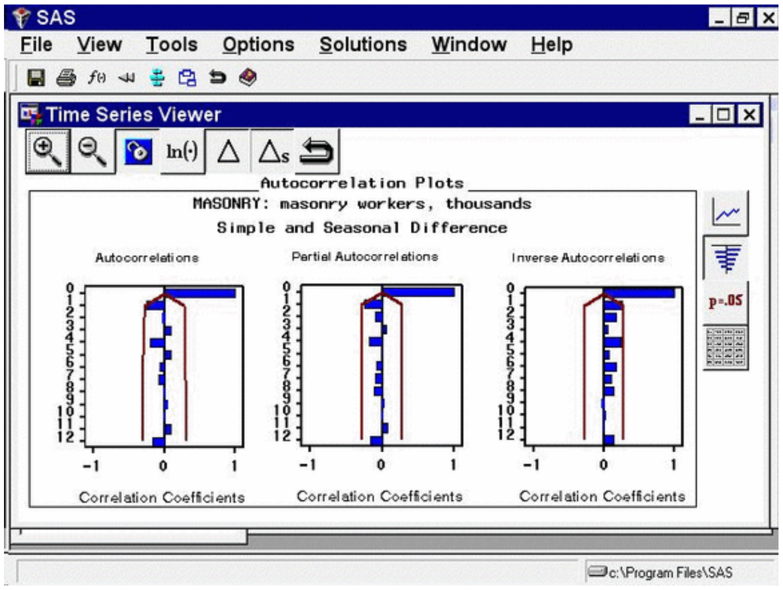
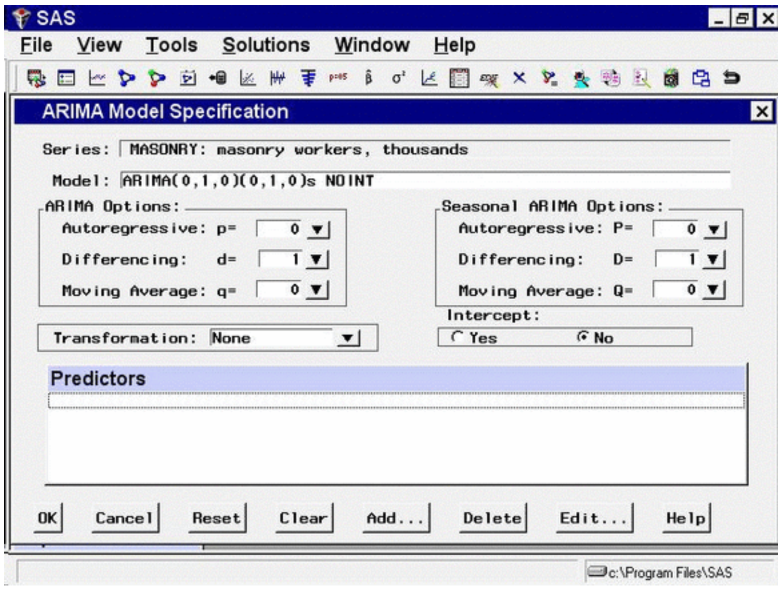


Figure 37.7. ACF Plot with Simple and Seasonal Differencing

Model Viewer Prediction Error Analysis

Leave the Time Series Viewer open for the remainder of this exercise. Drag it out of the way or push it to the background so that you can return to the Time Series Forecasting window. Select `Develop Models`, then click an empty part of the table to bring up the pop-up menu, and select `Fit ARIMA Model`. Define the ARIMA(0,1,0)(0,1,0)s model by selecting `1` for differencing under ARIMA Options, `1` for differencing under Seasonal ARIMA Options, and `No` for `Intercept`, as shown in Figure 37.8.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
ARIMA Options:
Seasonal ARIMA Options:
Predictors
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: MASONRY: masonry workers, thousands
Model: ARIMA(0,1,0)(0,1,0)s NOINT

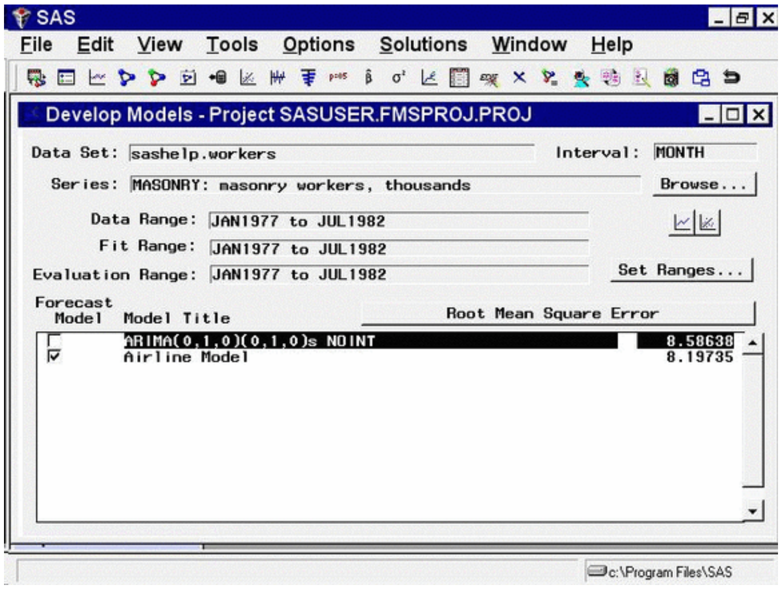
ARIMA Options		Seasonal ARIMA Options	
Autoregressive: p=	0	Autoregressive: P=	0
Differencing: d=	1	Differencing: D=	1
Moving Average: q=	0	Moving Average: Q=	0

Transformation: None

Intercept: Yes [✓ No]

Figure 37.8. Specifying the ARIMA(0,1,0)(0,1,0)s Model

When you select the `OK` button, the model is fit and you are returned to the Develop Models window. Click on an empty part of the table and choose `Fit Models from List` from the pop-up menu. Select `Airline Model` from the window. (Airline Model is a common name for the ARIMA(0,1,1)(0,1,1)s model, which is often used for seasonal data with a linear trend.) Select the `OK` button. Once the model has been fit, the table shows the two models and their root mean square errors. Notice that the Airline Model provides only a slight improvement over the differencing model, ARIMA(0,1,0)(0,1,0)s. Select the first row to highlight the differencing model, as shown in Figure 37.9.


SAS
File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...
c:\Program Files\SAS

Data Set: sashelp.workers Interval: MONTH
Series: MASONRY: masonry workers, thousands
Data Range: JAN1977 to JUL1982
Fit Range: JAN1977 to JUL1982
Evaluation Range: JAN1977 to JUL1982

Forecast Model	Model Title	Root Mean Square Error
☐	ARIMA(0,1,0)(0,1,0)s NOINT	8.58638
✓	Airline Model	8.19735

Figure 37.9. Selecting a Model

Now select the `View Selected Model Graphically` button, below the `Browse` button at the right side of the Develop Models window. The `Model Viewer` window appears, showing the actual data and model predictions for the MASONRY series. (Note that predicted values are missing for the first 13 observations due to simple and seasonal differencing.)

To examine the ACF plot for the model prediction errors, select the third icon from the top on the vertical toolbar. For this model, the prediction error ACF is the same as the ACF of the original data with first differencing and seasonal differencing applied. This differencing is apparent if you bring the Time Series Viewer back into view for comparison.

Return to the Develop Models Window by clicking on it or using the window pull-down menu or the Next Viewer toolbar icon. Select the second row of the table in the Develop Models window to highlight the Airline Model. The Model Viewer is automatically updated to show the prediction error ACF of the newly selected model, as shown in Figure 37.10.

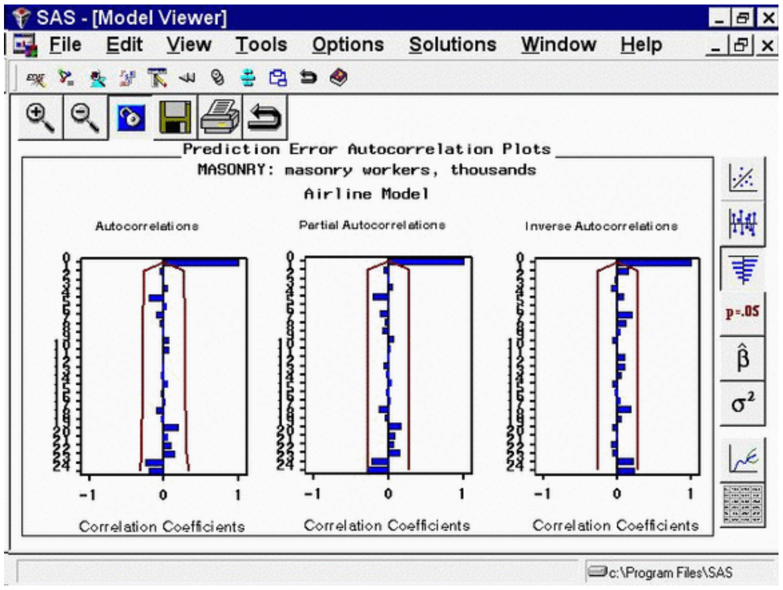
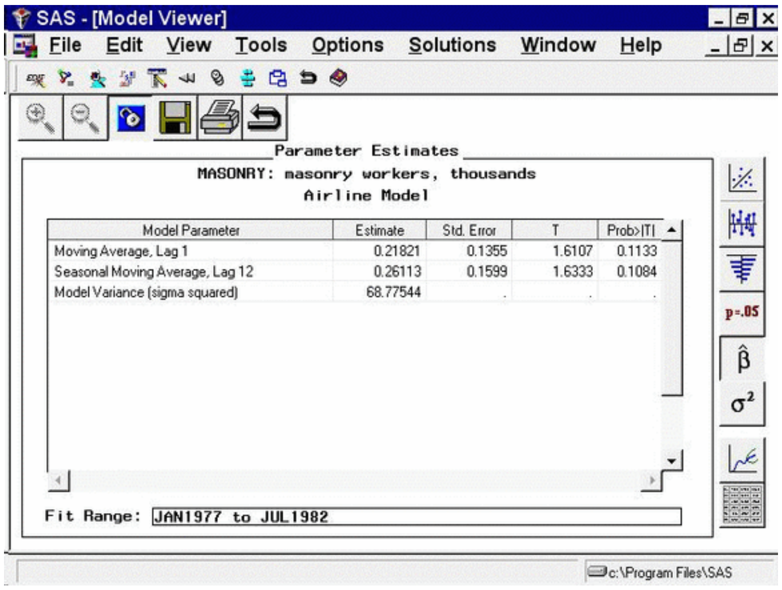


Figure 37.10. Prediction Error ACF Plot for the Airline Model

Another helpful tool available within the Model Viewer is the parameter estimates table. Select the fifth icon from the top of the vertical toolbar. The table gives the parameter estimates for the two moving average terms in the Airline Model, as well as the model residual variance, as shown in Figure 37.11.



Parameter Estimates

MASONRY: masonry workers, thousands
Airline Model

Model Parameter	Estimate	Std. Error	T	Prob>|T|
Moving Average, Lag 1	0.21821	0.1355	1.6107	0.1133
Seasonal Moving Average, Lag 12	0.26113	0.1599	1.6333	0.1084
Model Variance (sigma squared)	68.77544	.	.	.

Fit Range: JAN1977 to JUL1982

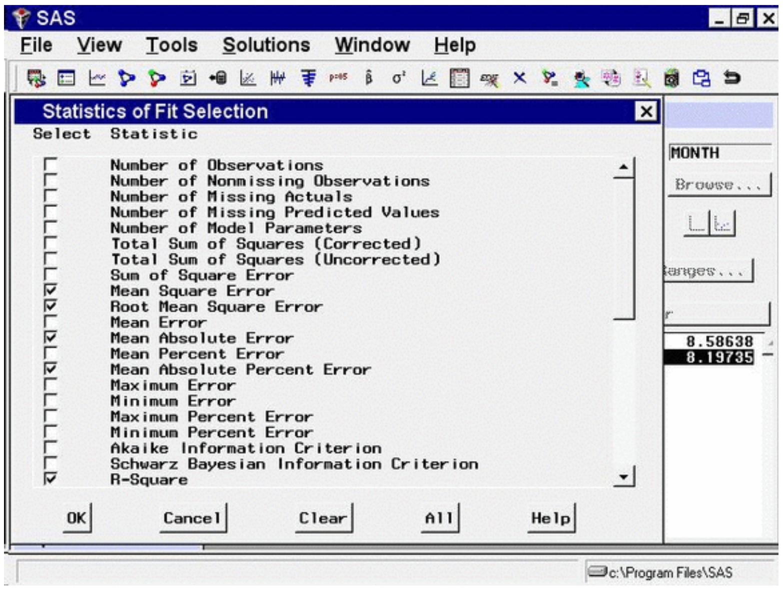
Figure 37.11. Parameter Estimates for the Airline Model

You can adjust the column widths in the table by dragging the vertical borders of the column titles with the mouse. Notice that neither of the parameter estimates is significantly different from zero at the .05 level of significance, since `Prob>|t|` is greater than .05. This suggests that the Airline Model should be discarded in favor of the more parsimonious differencing model, which has no parameters to estimate.

The Model Selection Criterion

Return to the Develop Models window (Figure 37.9) and notice the Root Mean Square Error button at the right side of the table banner. This is the model selection criterion–the statistic used by the system to select the best fitting model. So far in this example you have fit two models and have left the default criterion, root mean square error (RMSE), in effect. Because the Airline Model has the smaller value of this criterion, and because smaller values of the RMSE indicate better fit, the system has chosen this model as the forecasting model, indicated by the check box in the `Forecast Model` column.

The statistics available as model selection criteria are a subset of the statistics available for informational purposes. To access the entire set, select `Options` from the menu bar, and then select `Statistics of Fit`. The `Statistics of Fit Selection` window appears, as shown in Figure 37.12.



Select	Statistic
☐	Number of Observations
☐	Number of Nonmissing Observations
☐	Number of Missing Actuals
☐	Number of Missing Predicted Values
☐	Number of Model Parameters
☐	Total Sum of Squares (Corrected)
☐	Total Sum of Squares (Uncorrected)
☐	Sum of Square Error
☑	Mean Square Error
☑	Root Mean Square Error
☐	Mean Error
☑	Mean Absolute Error
☐	Mean Percent Error
☑	Mean Absolute Percent Error
☐	Maximum Error
☐	Minimum Error
☐	Maximum Percent Error
☐	Minimum Percent Error
☐	Akaike Information Criterion
☐	Schwarz Bayesian Information Criterion
☑	R-Square

Figure 37.12. Statistics of Fit

By default, five of the more well known statistics are selected. You can select and deselect statistics by clicking the check boxes in the left column. For this exercise, select `All`, and notice that all the check boxes become checked. Select the `OK` button to close the window. Now if you choose `Statistics of Fit` in the `Model Viewer` window, all of the statistics will be shown for the selected model.

To change the model selection criterion, click the `Root Mean Square Error` button or select `Options` from the menu bar and then select `Model Selection Criterion.` Notice that most of the statistics of fit are shown, but those which are not relevant to model selection, such as number of observations, are not shown. Select `Schwarz Bayesian Information Criterion` and click `OK`. Since this statistic puts a high penalty on models with larger numbers of parameters, the ARIMA(0,1,0)(0,1,0)s model comes out with the better fit.

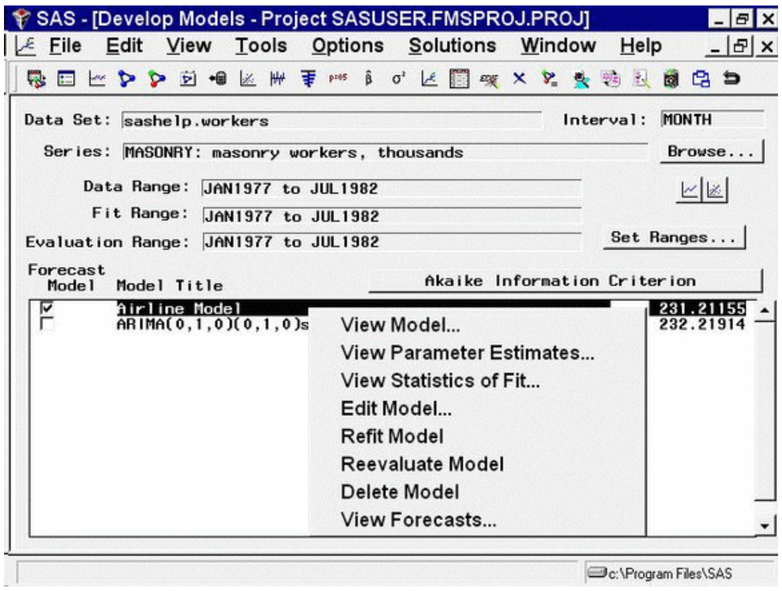
Notice that changing the selection criterion does not automatically select the model that is best according to that criterion. You can always choose the model you want to use for forecasts by selecting its check box in the `Forecast Model` column.

Now bring up the Model Selection Criterion window again and select `Akaike Information Criterion`. This statistic puts a lesser penalty on number of parameters, and the Airline Model comes out as the better fitting model.

Sorting and Selecting Models

Select `Sort Models` on the `Tools` pull-down menu or from the toolbar. This sorts the current list of fitted models by the current selection criterion. Although some selection criteria assign larger values to better fitting models (for example, R-square) while others assign smaller values to better fitting models, Sort Models always orders models with the best fitting model–in this case, the Airline Model–at the top of the list.

When you select a model in the table, its name and criterion value become highlighted, and actions that apply to that model become available. If your system supports a right mouse button, you can click it to invoke a pop-up menu, as shown in Figure 37.13.



Data Set: sashelp.workers   Interval: MONTH

Series: MASONRY: masonry workers, thousands

Data Range: JAN1977 to JUL1982

Fit Range: JAN1977 to JUL1982

Evaluation Range: JAN1977 to JUL1982

Forecast Model	Model Title	Akaike Information Criterion
✓	Airline Model	231.21155
	ARIMA(0,1,0)(0,1,0)s	232.21914

Figure 37.13. Right Mouse Button Pop-up Menu

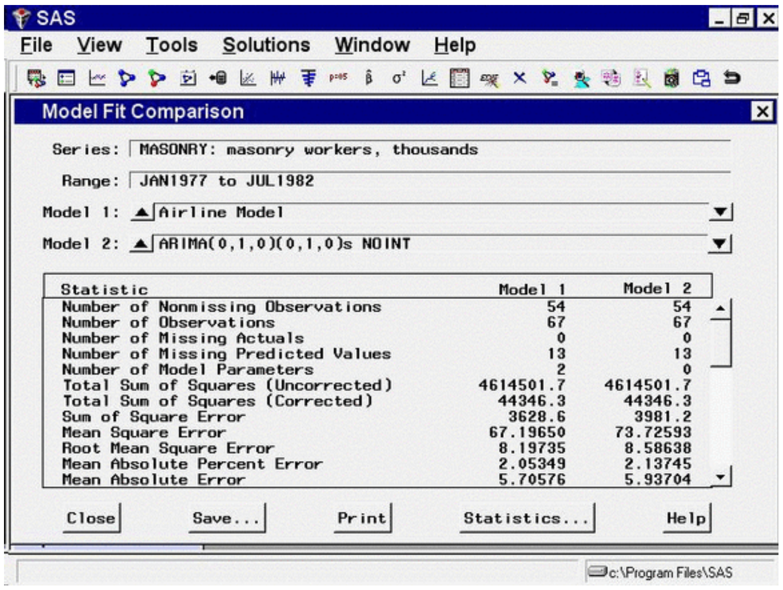
Whether or not you have a right mouse button, the same choices are available under `Edit` and `View` from the menu bar. If the model viewer has been invoked, it is automatically updated to show the selected model, unless you have unlinked the viewer using the `Link/Unlink` toolbar button.

Select the highlighted model in the table again. Notice that it is no longer highlighted. When no models are highlighted, the right mouse button pop-up menu changes, and items on the menu bar that apply to a selected model become grayed out. For example, you can choose `Edit` from the menu bar, but you can't choose the `Edit Model` or `Delete Model` selections unless you have highlighted a model in the table.

When you select the check box in the `Forecast Model` column of the table, the model in that row becomes the forecasting model. This is the model that will be used the next time forecasts are generated by choosing `View Forecasts`, or by using the `Produce Forecasts` window. Note that this forecasting model flag is automatically set when you use `Fit Automatic Model` or when you fit an individual model that fits better, using the current selection criterion, than the current forecasting model.

Comparing Models

Select `Tools` and `Compare Models` from the menu bar. This displays the `Model Fit Comparison` table, as shown in Figure 37.14.



Series: MASONRY: masonry workers, thousands

Range: JAN1977 to JUL1982

Model 1: Airline Model

Model 2: ARIMA(0,1,0)(0,1,0)s NOINT

```
Statistic                                      Model 1      Model 2
Number of Nonmissing Observations                   54           54
Number of Observations                              67           67
Number of Missing Actuals                            0            0
Number of Missing Predicted Values                  13           13
Number of Model Parameters                           2            0
Total Sum of Squares (Uncorrected)           4614501.7    4614501.7
Total Sum of Squares (Corrected)               44346.3      44346.3
Sum of Square Error                             3628.6       3981.2
Mean Square Error                             67.19650     73.72593
Root Mean Square Error                         8.19735      8.58638
Mean Absolute Percent Error                    2.05349      2.13745
Mean Absolute Error                            5.70576      5.93704
```

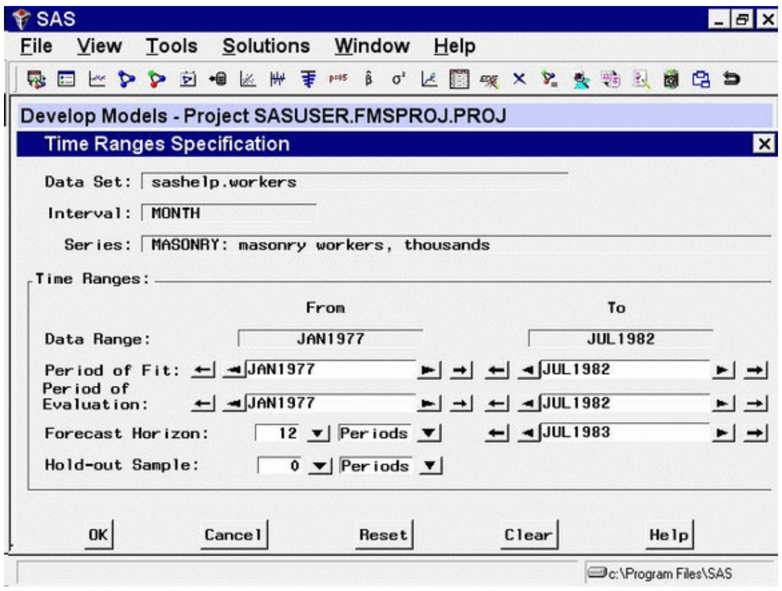
Figure 37.14. Model Comparison Window

The two models you have fit are shown as `Model 1` and `Model 2.` When there are more than two models, you can bring any two of them into the table by selecting the up and down arrows. In this way, it is easy to do pairwise comparisons on any number of models, looking at as many statistics of fit as you like. Since you previously chose to display all statistics of fit, all of them are shown in the comparison table. Use the vertical scroll bar to move through the list.

After you have examined the model comparison table, select the `Close` button to return to the `Develop Models` window.

Controlling the Period of Evaluation and Fit

Notice the three time ranges shown on the `Develop Models` window (Figure 37.9). The data range shows the beginning and ending dates of the MASONRY time series. The period of fit shows the beginning and ending dates of data used to fit the models. The period of evaluation shows the beginning and ending dates of data used to compute statistics of fit. By default, the fit and evaluate ranges are the same as the data range. To change these ranges, select the `Set Ranges` button, or select `Options` and `Time Ranges` from the menu bar. This brings up the `Time Ranges Specification` window, as shown in Figure 37.15.


SAS
File View Tools Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Time Ranges Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Data Set: sashelp.workers
Interval: MONTH
Series: MASONRY: masonry workers, thousands

Time Ranges:

	From		To
Data Range:	JAN1977		JUL1982
Period of Fit:	JAN1977		JUL1982
Period of Evaluation:	JAN1977		JUL1982
Forecast Horizon:	12	Periods	JUL1983
Hold-out Sample:	0	Periods	

Figure 37.15. Time Ranges Specification Window

For this example, suppose the early data in the series is unreliable, and you want to use the range June 1978 to the latest available for both model fitting and model evaluation. You can either type in JUN1978 in the `From` column for `Period of Fit` and `Period of Evaluation`, or you can advance these dates by clicking the right pointing arrows. The outer arrow advances the date by a large amount (in this case, by a year), and the inner arrow advances it by a single period (in this case, by a month). Once you have changed the `Period of Fit` and the `Period of Evaluation` to JUN1978 in the `From` column, select the `OK` button to return to the `Develop Models` window. Notice that these time ranges are updated at the top of the window, but the models already fit have not been affected. Your changes to the time ranges affect *subsequently fit* models.

Refitting and Reevaluating Models

If you fit the ARIMA(0,1,0)(0,1,0)s and Airline models again in the same way as before, they will be added to the model list, with the same names but with different values of the model selection criterion. Parameter estimates will be different, due to the new fit range, and statistics of fit will be different, due to the new evaluation range.

For this exercise, instead of specifying the models again, refit the existing models by selecting `Edit` from the menu bar and then selecting `Refit Models` and `All Models.` After the models have been refit, you should see the same two models listed in the table but with slightly different values for the selection criterion. The ARIMA (0,1,0)(0,1,0)s and Airline models have now been fit to the MASONRY series using data from June 1978 to July 1982, since this is the period of fit you specified. The statistics of fit have been computed for the period of evaluation, which was the same as the period of fit. If you had specified a period of evaluation different from the period of fit, the statistics would have been computed accordingly.

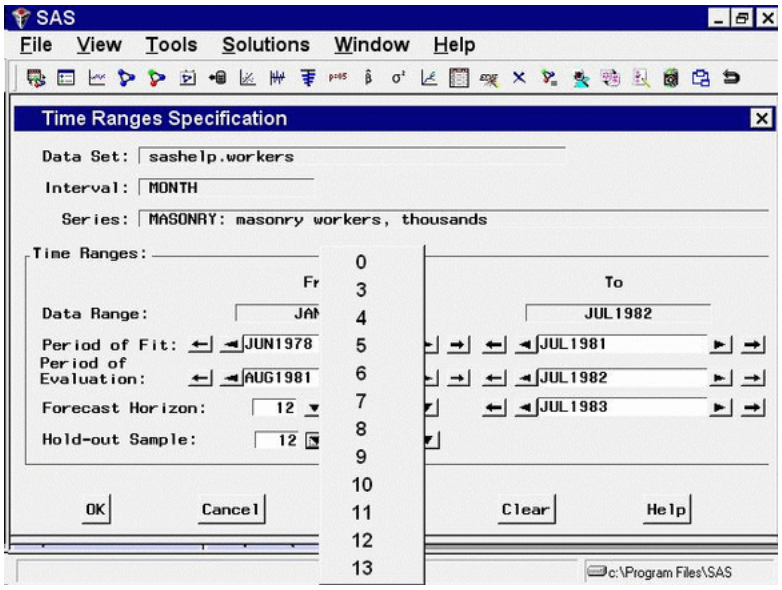
In practice, another common reason for refitting models is the availability of new data. For example, when data for a new month become available for a monthly series, you might add them to the input data set, then invoke the forecasting system, open the project containing models fit previously, and refit the models prior to generating new forecasts. Unless you specify the period of fit and period of evaluation in the `Time Ranges Specification` window, they default to the full data range of the series found in the input data set at the time of refitting.

If you prefer to apply previously fit models to revised data without refitting, use `Reevaluate Models` instead of `Refit Models`. This recomputes the statistics of fit using the current evaluation range, but does not re-estimate the model parameters.

Using Hold-out Samples

One important application of model fitting where the period of fit is different from the period of evaluation is the use of hold-out samples. With this technique of model evaluation, the period of fit ends at a time point before the end of the data series, and the remainder of the data are held out as a nonoverlapping period of evaluation. With respect to the period of fit, the hold-out sample is a period in the future, used to compare the forecasting accuracy of models fit to past data.

For this exercise, use a hold-out sample of 12 months. Bring up the `Time Ranges Specification` window again by selecting the `Set Ranges` button. Set `Hold-out Sample` to 12 using the combo box, as shown in Figure 37.16. You can also type in a value. To specify a hold-out sample period in different units, you can use the `Periods` combo box. In this case, it will allow you to select years as the unit, instead of periods.


SAS
File View Tools Solutions Window Help
Time Ranges Specification
OK
Cancel
Clear
Help
c:\Program Files\SAS

Time Ranges Specification

Data Set: sashelp.workers
Interval: MONTH
Series: MASONRY: masonry workers, thousands

Time Ranges:

	Fr	To
Data Range:	JAN	JUL1982
Period of Fit:	JUN1978	JUL1981
Period of Evaluation:	AUG1981	JUL1982
Forecast Horizon:	12	JUL1983
Hold-out Sample:	12	

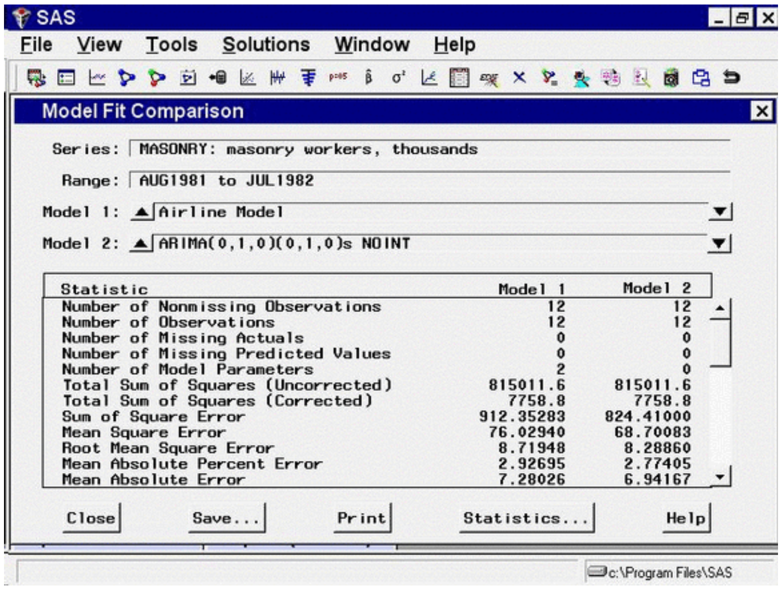
0
3
4
5
6
7
8
9
10
11
12
13

Figure 37.16. Specifying the Hold-out Sample Size

Notice that setting the holdout sample to 12 automatically sets the fit range to JUN1978–JUL1981 and the evaluation range to AUG1981–JUL1982. If you had set the period of fit and period of evaluation to these ranges, the hold-out sample would have been automatically set to 12 periods.

Select the `OK` button to return to the `Develop Models` window. Now refit the models again. Select `Tools` and `Compare Models` to compare the models now that they have been fit to the period June 1978 through July 1981 and evaluated for the hold-out sample period August 1981 through July 1982. Note that the fit statistics for the hold-out sample are based on one-step-ahead forecasts. (See *Statistics of Fit* in Chapter 41, "Forecasting Process Details,").

As shown in Figure 37.17, the ARIMA (0,1,0)(0,1,0)s model now seems to provide a better fit to the data than does the Airline model. It should be noted that the results can be quite different if you choose a different size hold-out sample.



```
Series:  MASONRY: masonry workers, thousands
Range:   AUG1981 to JUL1982
Model 1: Airline Model
Model 2: ARIMA(0,1,0)(0,1,0)s NOINT

Statistic                                  Model 1      Model 2
Number of Nonmissing Observations               12           12
Number of Observations                          12           12
Number of Missing Actuals                        0            0
Number of Missing Predicted Values               0            0
Number of Model Parameters                       2            0
Total Sum of Squares (Uncorrected)        815011.6     815011.6
Total Sum of Squares (Corrected)            7758.8       7758.8
Sum of Square Error                      912.35283    824.41000
Mean Square Error                         76.02940     68.70083
Root Mean Square Error                     8.71948      8.28860
Mean Absolute Percent Error                2.92695      2.77405
Mean Absolute Error                        7.28026      6.94167
```

Figure 37.17. Using 12 Month Hold-out Sample

Chapter 38
Using Predictor Variables

Chapter Contents

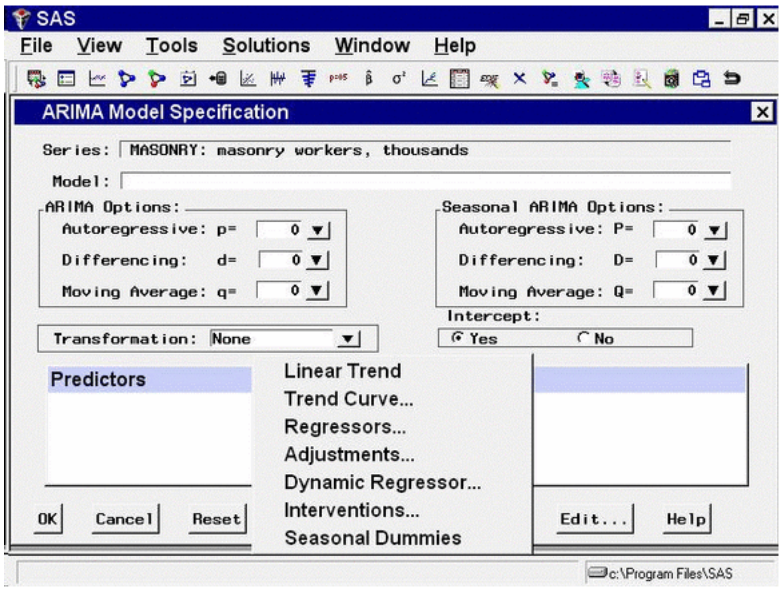
Chapter 38
Using Predictor Variables

Forecasting models predict the future values of a series using two sources of information: the past values of the series and the values of other time series variables. Other variables used to predict a series are called *predictor variables*.

Predictor variables that are used to predict the dependent series may be variables in the input data set, such as regressors and adjustment variables, or they can be special variables computed by the system as functions of time, such as trend curves, intervention variables, and seasonal dummies.

You can specify seven different types of predictors in forecasting models using the ARIMA Model or Custom Model Specification windows. You cannot specify predictor variables with the Smoothing Model Specification window.

Figure 38.1 shows the menu of options for adding predictors to an ARIMA model that is brought up by the Add button. The Add menu for the Custom Model Specification menu is similar.



Series: MASONRY: masonry workers, thousands
Model:
ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0
Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0
Transformation: None
Intercept: [✓ Yes] No

Figure 38.1. Add Predictors Menu

These types of predictors are as follows.

Linear Trend	adds a variable that indexes time as a predictor series. A straight line time trend is fit to the series by regression when you specify a linear trend.
Trend Curve	provides a menu of various functions of time that you can add to the model to fit nonlinear time trends. The Linear Trend option is a special case of the Trend Curve option for which the trend curve is a straight line.

Regressors	allows you to predict the series by regressing it on other variables in the data set.
Adjustments	allows you to specify other variables in the data set that supply adjustments to the forecast.
Dynamic Regressor	allows you to select a predictor variable from the input data set and specify a complex model for the way that the predictor variable affects the dependent series.
Interventions	allows you to model the effect of special events that "intervene" to change the pattern of the dependent series. Examples of intervention effects are strikes, tax increases, and special sales promotions.
Seasonal Dummies	adds seasonal indicator or "dummy" variables as regressors to model seasonal effects.

You can add any number of predictors to a forecasting model, and you can combine predictor variables with other model options.

The following sections explain these seven kinds of predictors in greater detail and provide examples of their use. The examples illustrate these different kinds of predictors using series in the SASHELP.USECON data set.

Select the `Develop Models` button from the main window. Select the data set SASHELP.USECON and select the series PETROL. Then select the `View Series Graphically` button from the Develop Models window. The plot of the example series PETROL appears as shown in Figure 38.2.

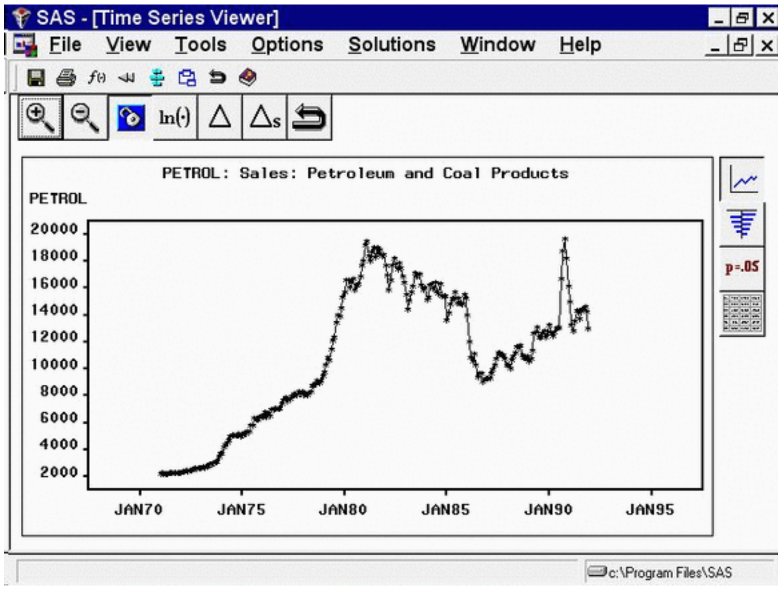
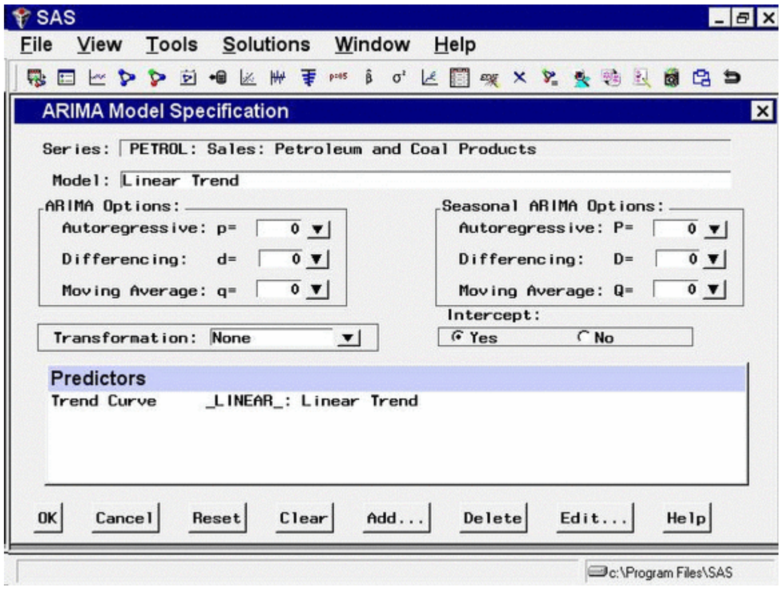


Figure 38.2. Sales of Petroleum and Coal

Linear Trend

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 38.1).

A linear trend is added to the Predictors list, as shown in Figure 38.3.



ARIMA Model Specification

Series: PETROL: Sales: Petroleum and Coal Products

Model: Linear Trend

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Transformation: None

Intercept: [✓ Yes] No

Predictors
Trend Curve _LINEAR_: Linear Trend

Figure 38.3. Linear Trend Predictor Specified

The description for the linear trend item shown in the Predictors list has the following meaning. The first part of the description, Trend Curve, describes the type of predictor. The second part, _LINEAR_, gives the variable name of the predictor series. In this case, the variable is a time index that the system computes. This variable will be included in the output forecast data set. The final part, Linear Trend, describes the predictor.

Notice that the model you have specified consists only of the time index regressor _LINEAR_ and an intercept. Although this window is normally used to specify ARIMA models, in this case no ARIMA model options are specified, and the model is a simple regression on time.

Select the `OK` button. The Linear Trend model is fit and added to the model list in the Develop Models window.

Now bring up the Model Viewer using the `View Model Graphically` icon or the `Model Predictions` item under the `View` pull-down menu or toolbar. This displays a plot of the model predictions and actual series values, as shown in Figure 38.4. The predicted values lie along the least squares trend line.

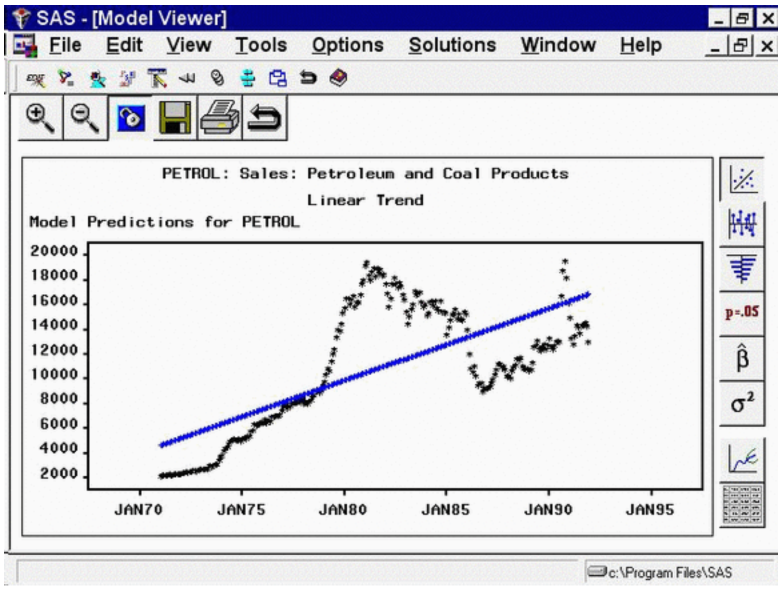
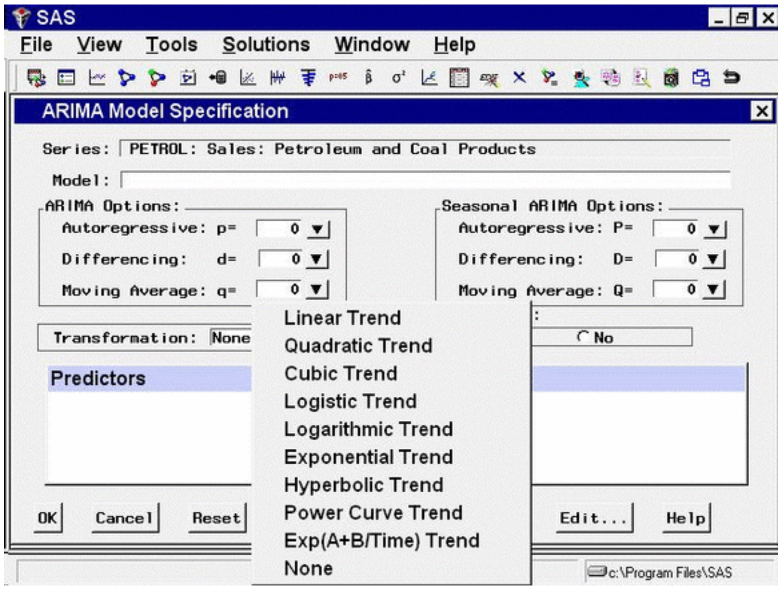


Figure 38.4. Linear Trend Model

Time Trend Curves

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select Add and then select `Trend Curve` from the menu (shown in Figure 38.1). A menu of different kinds of trend curves is displayed, as shown in Figure 38.5.



Series: PETROL: Sales: Petroleum and Coal Products

Model:

ARIMA Options:
- Autoregressive: p= 0
- Differencing: d= 0
- Moving Average: q= 0

Seasonal ARIMA Options:
- Autoregressive: P= 0
- Differencing: D= 0
- Moving Average: Q= 0

Transformation: None

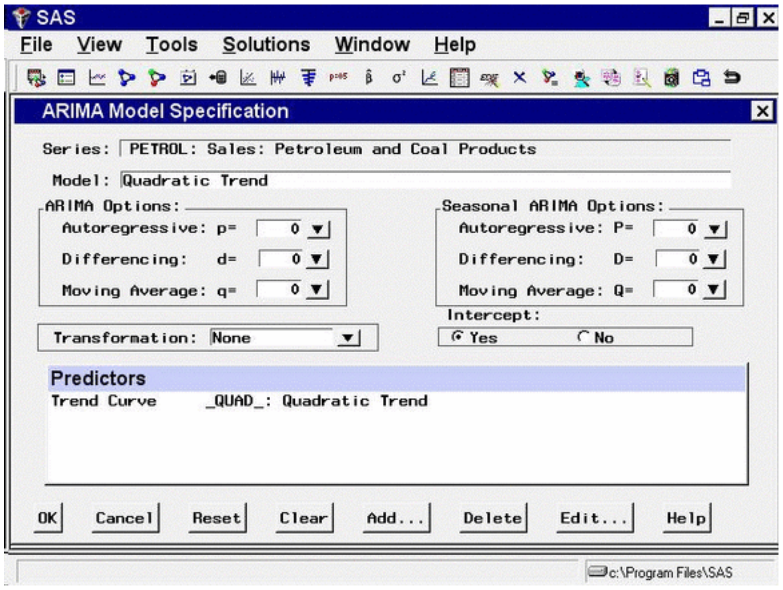
No

Predictors

Figure 38.5. Time Trend Curves Menu

These trend curves work in a similar way as the Linear Trend option (which is a special case of a trend curve and one of the choices on the menu), but with the Trend Curve menu you have a choice of various nonlinear time trends.

Select `Quadratic Trend`. This adds a quadratic time trend to the Predictors list, as shown in Figure 38.6.



SAS

File View Tools Solutions Window Help

ARIMA Model Specification

Series: PETROL: Sales: Petroleum and Coal Products

Model: Quadratic Trend

ARIMA Options: Autoregressive: p= 0; Differencing: d= 0; Moving Average: q= 0

Seasonal ARIMA Options: Autoregressive: P= 0; Differencing: D= 0; Moving Average: Q= 0

Transformation: None

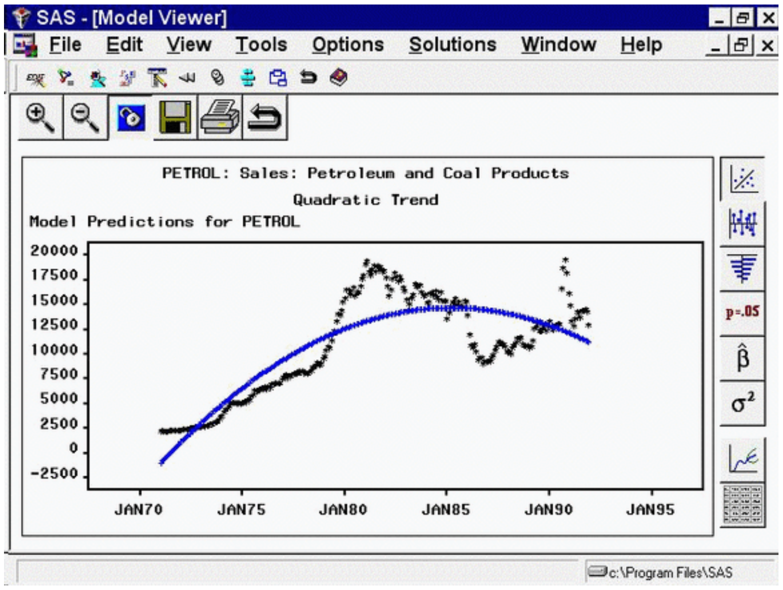
Intercept: [✓ Yes] No

Predictors

Trend Curve _QUAD_: Quadratic Trend

Figure 38.6. Quadratic Trend Specified

Now select the `OK` button. The quadratic trend model is fit and added to the list of models in the Develop Models window. The Model Viewer displays a plot of the quadratic trend model, as shown in Figure 38.7.



SAS - [Model Viewer]

File Edit View Tools Options Solutions Window Help

PETROL: Sales: Petroleum and Coal Products

Quadratic Trend

Model Predictions for PETROL

Figure 38.7. Quadratic Trend Model

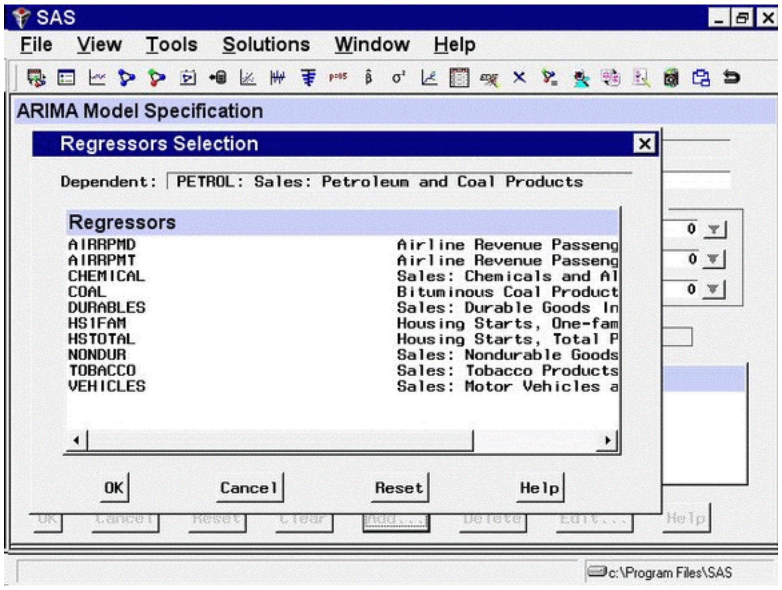
This curve does not fit the PETROL series very well, but the View Model plot illustrates how time trend models work. You may want to experiment with different trend models to see what the different trend curves look like.

Some of the trend curves require transforming the dependent series. When you specify one of these curves, a notice is displayed reminding you that a transformation is needed, and the Transformation field is automatically filled in. Therefore, you cannot control the Transformation specification when some kinds of trend curves are specified.

See the section "Time Trend Curves" in Chapter 41, “Forecasting Process Details,” for more information about the different trend curves.

Regressors

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Regressors` from the menu (shown in Figure 38.1). This displays the `Regressors Selection` window, as shown in Figure 38.8. This window allows you to select any number of other series in the input data set as regressors to predict the dependent series.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
Regressors Selection
OK
Cancel
Reset
Help
c:\Program Files\SAS

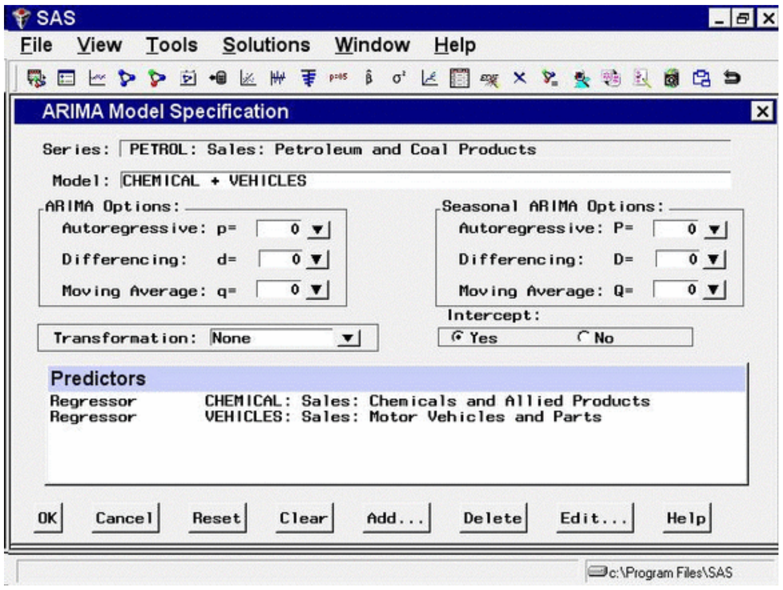
Dependent: PETROL: Sales: Petroleum and Coal Products

Regressors

AIRRPMD	Airline Revenue Passeng
AIRRPMT	Airline Revenue Passeng
CHEMICAL	Sales: Chemicals and Al
COAL	Bituminous Coal Product
DURABLES	Sales: Durable Goods In
HS1FAM	Housing Starts, One-fam
HSTOTAL	Housing Starts, Total P
NONDUR	Sales: Nondurable Goods
TOBACCO	Sales: Tobacco Products
VEHICLES	Sales: Motor Vehicles a

Figure 38.8. Regressors Selection Window

For this example, select `CHEMICAL, Sales: Chemicals and Allied Products`, and `VEHICLES, Sales: Motor Vehicles and Parts`. (Note: You do not need to use the CTRL key when selecting more than one regressor.) Then select the `OK` button. The two variables you selected are added to the Predictors list as regressor type predictors, as shown in Figure 38.9.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Model: CHEMICAL + VEHICLES

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Intercept: [✓ Yes] No

Transformation: None

Predictors

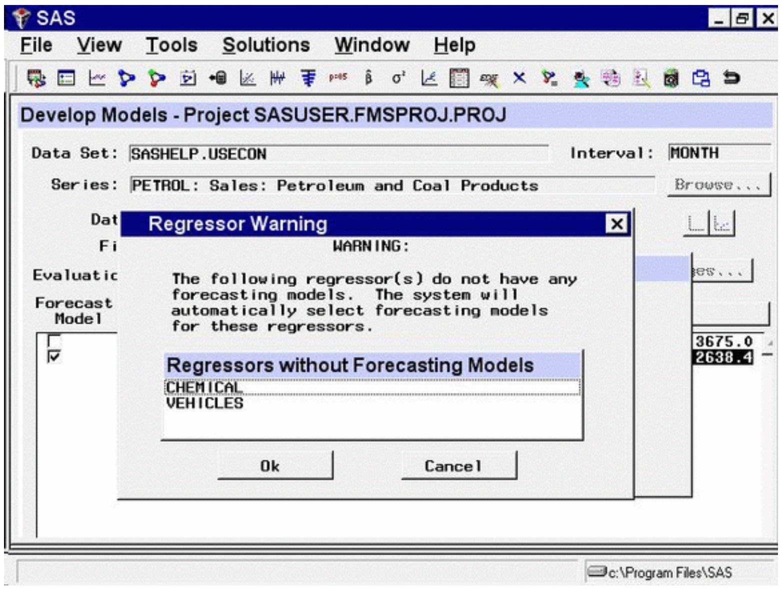
```
Regressor        CHEMICAL: Sales: Chemicals and Allied Products
Regressor        VEHICLES: Sales: Motor Vehicles and Parts
```

Figure 38.9. Regressors Selected

You must have forecasts of the future values of the regressor variables in order to use them as predictors. To do this, you can specify a forecasting model for each regressor, have the system automatically select forecasting models for the regressors, or supply predicted future values for the regressors in the input data set.

Even if you have supplied future values for a regressor variable, the system requires a forecasting model for the regressor. Future values that you supply in the input data set will take precedence over predicted values from the regressor's forecasting model when the system computes the forecasts for the dependent series.

Select the `OK` button. The system starts to fit the regression model but then stops and displays a warning that the regressors that you selected do not have forecasting models, as shown in Figure 38.10.



Develop Models - Project SASUSER.FMSPROJ.PROJ

Data Set: SASHELP.USECON Interval: MONTH

Series: PETROL: Sales: Petroleum and Coal Products

Regressor Warning

WARNING:

The following regressor(s) do not have any forecasting models. The system will automatically select forecasting models for these regressors.

Regressors without Forecasting Models

CHEMICAL
VEHICLES

Figure 38.10. Regressors Needing Models Warning

If you want the system to create forecasting models automatically for the regressor variables using the automatic model selection process, select the `OK` button. If not, you can select the `Cancel` button to abort fitting the regression model.

For this example, select the `OK` button. The system now performs the automatic model selection process for CHEMICAL and VEHICLES. The selected forecasting models for CHEMICAL and VEHICLES are added to the model lists for those series. If you switch the current time series in the Develop Models window to CHEMICAL or VEHICLES, you will see the model that the system selected for that series.

Once forecasting models have been fit for all regressors, the system proceeds to fit the regression model for PETROL. The fitted regression model is added to the model list displayed in the Develop Models window.

Adjustments

An *adjustment* predictor is a variable in the input data set that is used to adjust the forecast values produced by the forecasting model. Unlike a regressor, an adjustment variable does not have a regression coefficient. No model fitting is performed for adjustments. Nonmissing values of the adjustment series are simply added to the model prediction for the corresponding period. Missing adjustment values are ignored. If you supply adjustment values for observations within the period of fit, the adjustment values are subtracted from the actual values, and the model is fit to these adjusted values.

To add adjustments, select `Add` and then select `Adjustments` from the pop-up menu (shown in Figure 38.1). This displays the `Adjustments Selection` window. The Adjustments Selection window functions the same as the Regressor Selection window (which is shown in Figure 38.8). You can select any number of adjustment variables as predictors.

Unlike regressors, adjustments do not require forecasting models for the adjustment variables. If a variable that is used as an adjustment does have a forecasting model fit to it, the adjustment variable's forecasting model is ignored when the variable is used as an adjustment.

You can use forecast adjustments to account for expected future events that have no precedent in the past and so cannot be modeled by regression. For example, suppose you are trying to forecast the sales of a product, and you know that a special promotional campaign for the product is planned during part of the period you want to forecast. If such sales promotion programs have been frequent in the past, then you can record the past and expected future level of promotional efforts in a variable in the data set and use that variable as a regressor in the forecasting model.

However, if this is the first sales promotion of its kind for this product, you have no way to estimate the effect of the promotion from past data. In this case, the best you can do is to make an educated guess at the effect the promotion will have and add that guess to what your forecasting model would predict in the absence of the special sales campaign.

Adjustments are also useful for making judgmental alterations to forecasts. For example, suppose you have produced forecast sales data for the next 12 months. Your supervisor believes that the forecasts are too optimistic near the end and asks you to prepare a forecast graph in which the numbers that you have forecast are reduced by 1000 in the last three months. You can accomplish this task by editing the input data set so that it contains observations for the actual data range of sales plus 12 additional observations for the forecast period, and a new variable called, for example, ADJUSTMENT. The variable ADJUSTMENT contains the value 1000 for the last three observations, and is missing for all other observations. You fit the same model previously selected for forecasting using the `ARIMA Model Specification` or `Custom Model Specification` window, but with an adjustment added using the variable ADJUSTMENT. Now when you graph the forecasts using the Model Viewer, the last three periods of the forecast are reduced by 1000. The confidence limits are unchanged, which helps draw attention to the fact that the adjustments to the forecast deviate from what would be expected statistically.

Dynamic Regressor

Selecting `Dynamic Regressor` from the `Add Predictors` menu (shown in Figure 38.1) allows you to specify a complex time series model of the way that a predictor variable influences the series that you are forecasting.

When you specify a predictor variable as a simple regressor, only the current period value of the predictor effects the forecast for the period. By specifying the predictor with the Dynamic Regression option, you can use past values of the predictor series, and you can model effects that take place gradually.

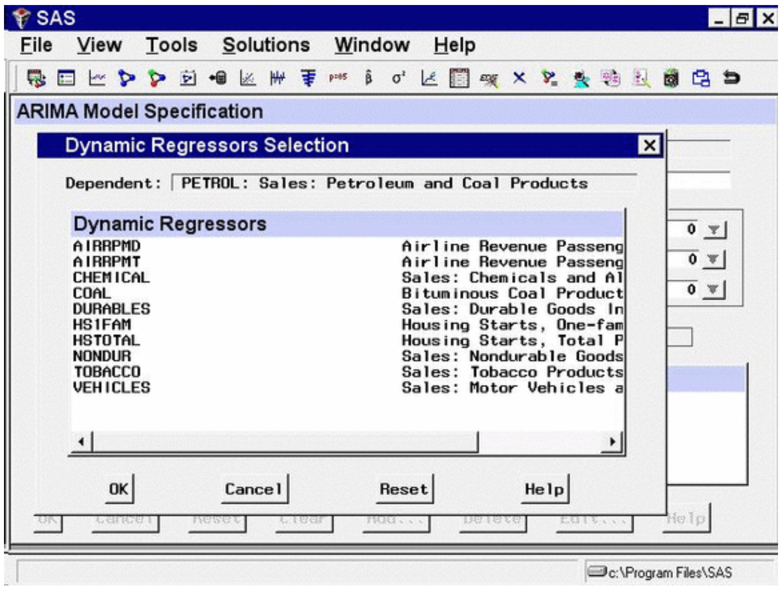
Dynamic regression models are an advanced feature that you are unlikely to find useful unless you have studied the theory of statistical time series analysis. You may want to skip this section if you are not trained in time series modeling.

The term *dynamic regression* was introduced by Pankratz (1991) and refers to what Box and Jenkins (1976) named *transfer function models*. In dynamic regression, you have a time series model, similar to an ARIMA model, that predicts how changes in the predictor series affect the dependent series over time.

The dynamic regression model relates the predictor variable to the expected value of the dependent series in the same way that an ARIMA model relates the fluctuations of the dependent series about its conditional mean to the random error term (which is also called the innovation series). Refer to "*Forecasting with Dynamic Regression Models*" (Pankratz, 1991) and) "*Time Series Analysis: Forecasting and Control*" (Box and Jenkins, 1976 for more information on dynamic regression or transfer function models. See also Chapter 11, "The ARIMA Procedure."

From the Develop Models window, select `Fit ARIMA Model.` From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 38.1).

Now select `Add` and select `Dynamic Regressor`. This displays the `Dynamic Regressors Selection` window, as shown in Figure 38.11.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
Dynamic Regressors Selection
OK
Cancel
Reset
Help
c:\Program Files\SAS

Dependent: PETROL: Sales: Petroleum and Coal Products

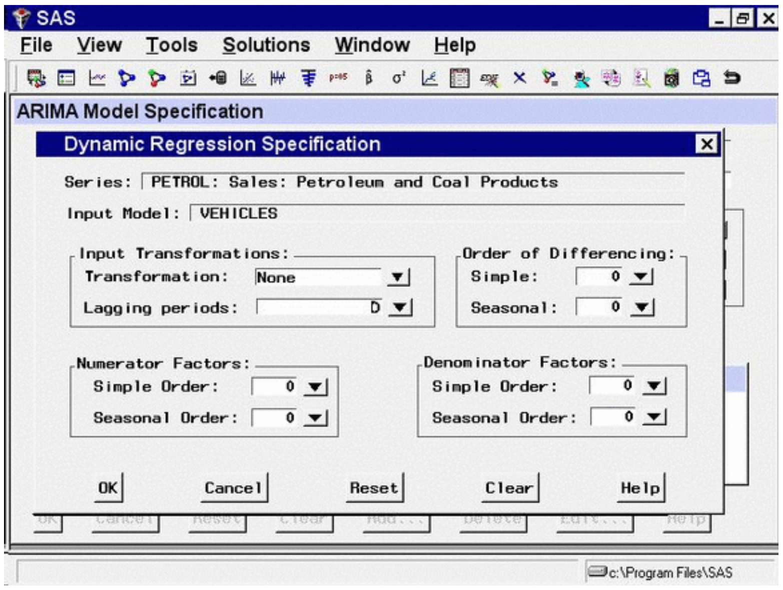
Dynamic Regressors

```
AIRRPMD      Airline Revenue Passeng
AIRRPMT      Airline Revenue Passeng
CHEMICAL     Sales: Chemicals and Al
COAL         Bituminous Coal Product
DURABLES     Sales: Durable Goods In
HS1FAM       Housing Starts, One-fam
HSTOTAL      Housing Starts, Total P
NONDUR       Sales: Nondurable Goods
TOBACCO      Sales: Tobacco Products
VEHICLES     Sales: Motor Vehicles a
```

Figure 38.11. Dynamic Regressors Selection Window

You can select only one predictor series when specifying a dynamic regression model. For this example, select `VEHICLES, Sales: Motor Vehicles and Parts`. Then select the `OK` button.

This displays the `Dynamic Regression Specification` window, as shown in Figure 38.12.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
Dynamic Regression Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products

Input Model: VEHICLES

Input Transformations:

Transformation: None

Lagging periods: 0

Order of Differencing:

Simple: 0

Seasonal: 0

Numerator Factors:

Simple Order: 0

Seasonal Order: 0

Denominator Factors:

Simple Order: 0

Seasonal Order: 0

Figure 38.12. Dynamic Regression Specification Window

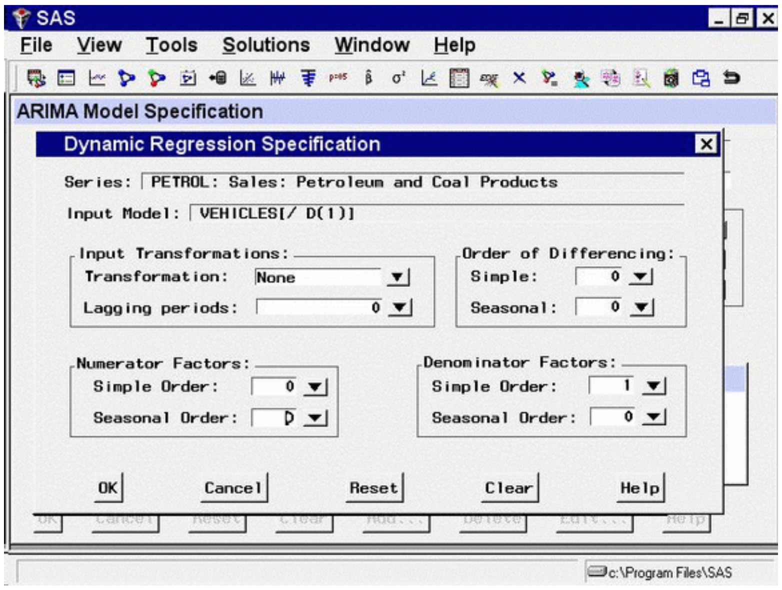
This window consists of four parts. The `Input Transformations` fields allow you to transform or lag the predictor variable. For example, you might use the lagged logarithm of the variable as the predictor series.

The `Order of Differencing` fields allow you to specify simple and seasonal differencing of the predictor series. For example, you might use changes in the predictor variable instead of the variable itself as the predictor series.

The `Numerator Factors` and `Denominator Factors` fields allow you to specify the orders of simple and seasonal numerator and denominator factors of the transfer function.

Simple regression is a special case of dynamic regression in which the dynamic regression model consists of only a single regression coefficient for the current value of the predictor series. If you select the `OK` button without specifying any options in the Dynamic Regression Specification window, a simple regressor will be added to the model.

For this example, use the `Simple Order` combo box for `Denominator Factors` and set its value to 1. The window now appears as shown in Figure 38.13.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
Dynamic Regression Specification
OK Cancel Reset Clear Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Input Model: VEHICLES[/ D(1)]

Input Transformations:
Transformation: None
Lagging periods: 0

Order of Differencing:
Simple: 0
Seasonal: 0

Numerator Factors:
Simple Order: 0
Seasonal Order: 0

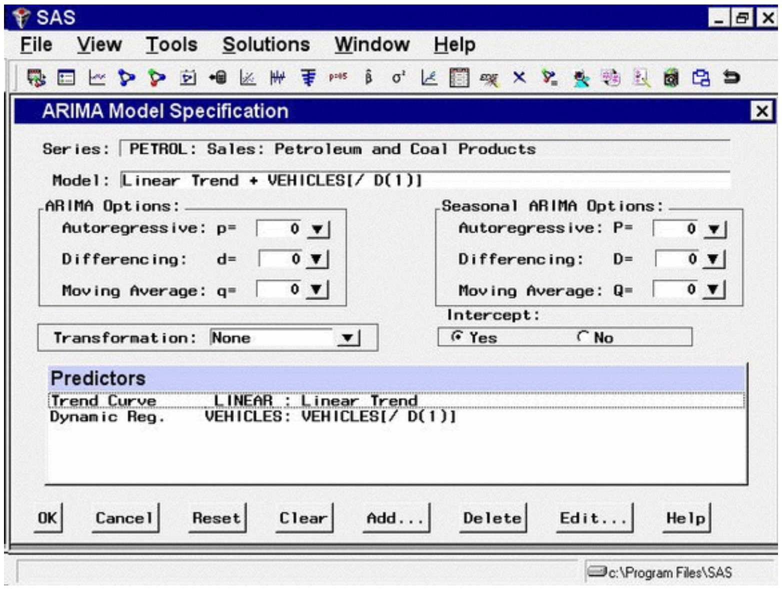
Denominator Factors:
Simple Order: 1
Seasonal Order: 0

Figure 38.13. Distributed Lag Regression Specified

This model is equivalent to regression on an exponentially weighted infinite distributed lag of VEHICLES (in the same way an MA(1) model is equivalent to single exponential smoothing).

Select the `OK` button to add the dynamic regressor to the model predictors list.

In the ARIMA Model Specification window, the Predictors list should now contain two items, a linear trend and a dynamic regressor for VEHICLES, as shown in Figure 38.14.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Model: Linear Trend + VEHICLES[/ D(1)]

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Intercept: [✓ Yes] No

Transformation: None

Predictors
```
Trend Curve      LINEAR : Linear Trend
Dynamic Reg.    VEHICLES: VEHICLES[/ D(1)]
```

Figure 38.14. Dynamic Regression Model

This model is a multiple regression of PETROL on a time trend variable and an infinite distributed lag of VEHICLES. Select the `OK` button to fit the model.

As with simple regressors, if VEHICLES does not already have a forecasting model, an automatic model selection process is performed to find a forecasting model for VEHICLES before the dynamic regression model for PETROL is fit.

Interventions

An *intervention* is a special indicator variable, computed automatically by the system, that identifies time periods affected by unusual events that influence or intervene in the normal path of the time series you are forecasting. When you add an intervention predictor, the indicator variable of the intervention is used as a regressor, and the impact of the intervention event is estimated by regression analysis.

To add an intervention to the Predictors list, you must use the Intervention Specification window to specify the time or times that the intervening event took place and to specify the type of intervention. You can add interventions either through the `Interventions` item of the `Add` action or by selecting `Tools` from the menu bar and then selecting `Define Interventions.`

Intervention specifications are associated with the series. You can specify any number of interventions for each series, and once you define interventions you can select them for inclusion in forecasting models. If you select the `Include Interventions` option in the `Options` pull-down menu, any interventions that you have previously specified for a series are automatically added as predictors to forecasting models for the series.

From the Develop Models window, invoke the series viewer by selecting the `View Series Graphically` icon or `Series` under the `View` pull-down menu. This displays the Time Series Viewer, as was shown in Figure 38.2.

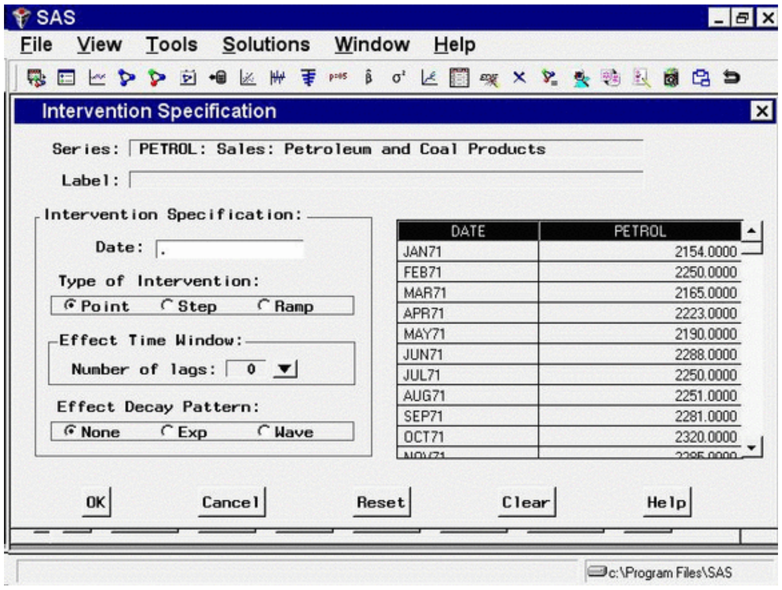
Note that the trend in the PETROL series shows several clear changes in direction. The upward trend in the first part of the series reverses in 1981. There is a sharp drop in the series towards the end of 1985, after which the trend is again upwardly sloped. Finally, in 1991 the series takes a sharp upward excursion but quickly returns to the trend line.

You may have no idea what events caused these changes in the trend of the series, but you can use these patterns to illustrate the use of intervention predictors. To do this, you fit a linear trend model to the series, but modify that trend line by adding intervention effects to model the changes in trend you observe in the series plot.

The Intervention Specification Window

From the Develop Models window, select `Fit ARIMA` model. From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 38.1).

Select `Add` again and then select `Interventions.` If you have any interventions already defined for the series, this selection displays the `Interventions for Series` window. However, since you have not previously defined any interventions, this list is empty. Therefore, the system assumes that you want to add an intervention and displays the `Intervention Specification` window instead, as shown in Figure 38.15.


SAS
File View Tools Solutions Window Help
Intervention Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products

Label:

Intervention Specification:

Date: .

Type of Intervention: [✓ Point] Step Ramp

Effect Time Window: Number of lags: 0

Effect Decay Pattern: [✓ None] Exp Wave

DATE	PETROL
JAN71	2154.0000
FEB71	2250.0000
MAR71	2165.0000
APR71	2223.0000
MAY71	2190.0000
JUN71	2288.0000
JUL71	2250.0000
AUG71	2251.0000
SEP71	2281.0000
OCT71	2320.0000

Figure 38.15. Interventions Specification Window

The top of the Intervention Specification window shows the current series and the label for the new intervention (initially blank). At the right side of the window is a scrollable table showing the values of the series. This table helps you locate the dates of the events you want to model.

At the left of the window is an area titled `Intervention Specification` that contains the options for defining the intervention predictor. The `Date` field specifies the time that the intervention occurs. You can type a date value in the `Date` field, or you can set the Date value by selecting a row from the table of series values at the right side of the window.

The area titled `Type of Intervention` controls the kind of indicator variable constructed to model the intervention effect. You can specify the following kinds of interventions:

Point	is used to indicate an event that occurs in a single time period. An example of a point event is a strike that shuts down production for part of a time period. The value of the intervention's indicator variable is zero except for the date specified.
Step	is used to indicate a continuing event that changes the level of the series. An example of a step event is a change in the law, such as a tax rate increase. The value of the intervention's indicator variable is zero before the date specified and 1 thereafter.
Ramp	is used to indicate a continuing event that changes the trend of the series. The value of the intervention's indicator variable is zero before the date specified, and it increases linearly with time thereafter.

The areas titled `Effect Time Window` and `Effect Decay Pattern` specify how to model the effect that the intervention has on the dependent series. These options are not used for simple interventions, they will be discussed later in this chapter.

Specifying a Trend Change Intervention

In the Time Series Viewer window position the mouse over the highest point in 1981 and select the point. This displays the data value, 19425, and date, February 1981, of that point in the upper-right corner of the Time Series Viewer, as shown in Figure 38.16.

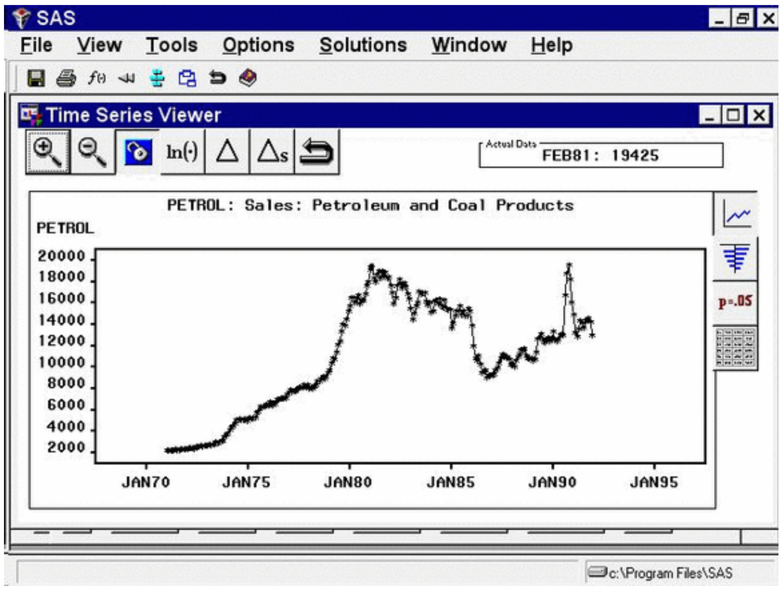
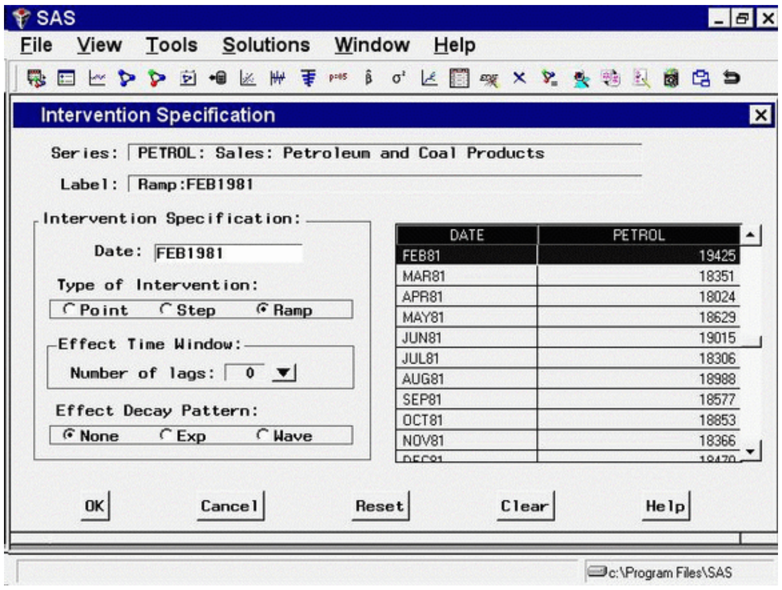


Figure 38.16. Identifying the Turning Point

Now that you know the date that the trend reversal occurred, enter that date in the `Date` field of the Intervention Specification window. Select `Ramp` as the type of intervention. The window should now appear as shown in Figure 38.17.

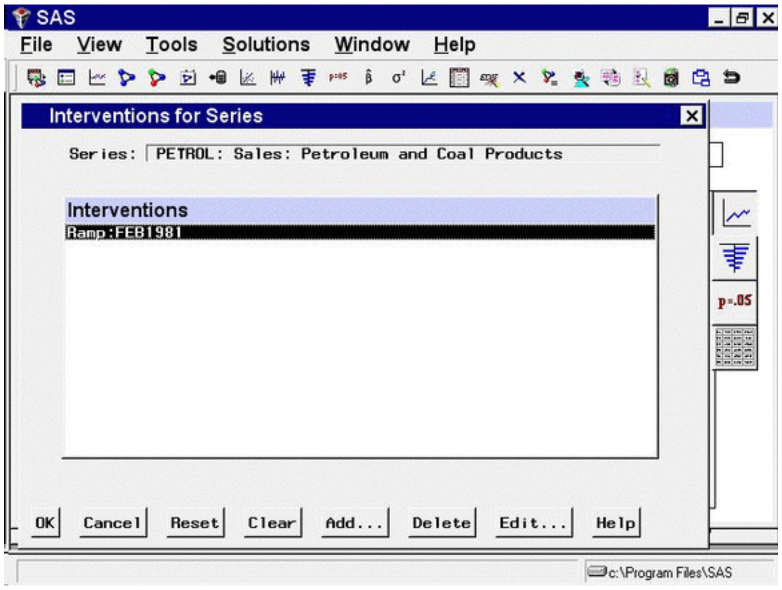

SAS
File View Tools Solutions Window Help
Intervention Specification
OK Cancel Reset Clear Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Label: Ramp:FEB1981
Intervention Specification:
Date: FEB1981
Type of Intervention: ( ) Point ( ) Step [✓ Ramp]
Effect Time Window:
Number of lags: 0
Effect Decay Pattern: [✓ None] ( ) Exp ( ) Wave

DATE	PETROL
FEB81	19425
MAR81	18351
APR81	18024
MAY81	18629
JUN81	19015
JUL81	18306
AUG81	18988
SEP81	18577
OCT81	18853
NOV81	18366

Figure 38.17. Ramp Intervention Specified

Select the `OK` button. This adds the intervention to the list of interventions for the PETROL series, and returns you to the `Interventions for Series` window, as shown in Figure 38.18.


SAS
File View Tools Solutions Window Help
Interventions for Series
OK Cancel Reset Clear Add... Delete Edit... Help
p=.05
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Interventions
Ramp:FEB1981

Figure 38.18. Interventions for Series Window

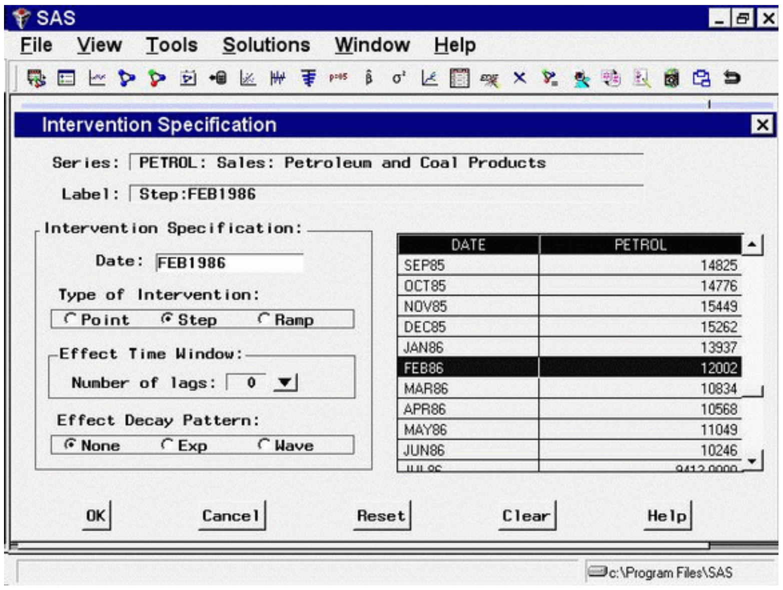
This window allows you to select interventions for inclusion in the forecasting model. Since you need to define other interventions, select the `Add` button. This returns you to the Intervention Specification window (shown in Figure 38.15).

Specifying a Level Change Intervention

Now add an intervention to account for the drop in the series in late 1985. You can locate the date of this event by selecting points in the Time Series Viewer plot or by scrolling through the data values table in the Interventions Specification window. Use the latter method so that you can see how this works.

Scrolling through the table, you see that the drop was from 15262 in December 1985, to 13937 in January 1986, to 12002 in February, to 10834 in March. Since the drop took place over several periods, you could use another ramp type intervention. However, this example represents the drop as a sudden event using a step intervention and uses February 1986 as the approximate time of the drop.

Select the table row for February 1986 to set the `Date` field. Select `Step` as the intervention type. The window should now appear as shown in Figure 38.19.


SAS
File View Tools Solutions Window Help
Intervention Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products

Label: Step:FEB1986

Intervention Specification:

Date: FEB1986

Type of Intervention: Point [✓ Step] Ramp

Effect Time Window:

Number of lags: 0

Effect Decay Pattern: [✓ None] Exp Wave

DATE	PETROL
SEP85	14825
OCT85	14776
NOV85	15449
DEC85	15262
JAN86	13937
FEB86	12002
MAR86	10834
APR86	10568
MAY86	11049
JUN86	10246

Figure 38.19. Step Intervention Specified

Select the `OK` button to add this intervention to the list for the series.

Since the trend reverses again after the drop, add a ramp intervention for the same date as the step intervention. Select `Add` from the Interventions for Series window. Enter FEB86 in the `Date` field, select `Ramp,` and then select the `OK` button.

Modeling Complex Intervention Effects

You have now defined three interventions to model the changes in trend and level. The excursion near the end of the series remains to be dealt with.

Select `Add` from the Interventions for Series window. Scroll through the data values and select the date on which the excursion began, August 1990. Leave the intervention type as Point.

The pattern of the series from August 1990 through January 1991 is more complex than a simple shift in level or trend. For this pattern, you need a complex intervention model for an event that causes a sharp rise followed by a rapid return to the previous trend line. To specify this model, use the `Effect Time Window` and `Effect Decay Rate` options.

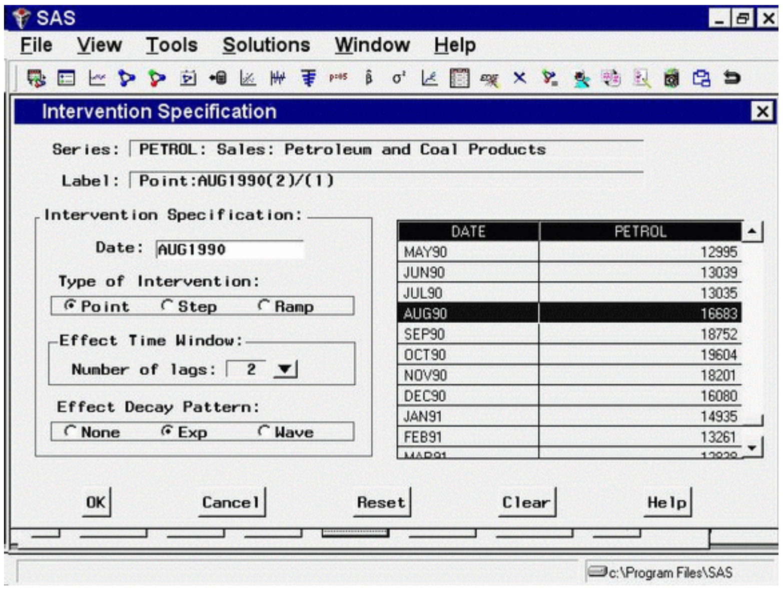
The `Effect Time Window` option controls the number of lags of the intervention's indicator variable used to model the effect of the intervention on the dependent series. For a simple intervention, the number of lags is zero, which means that the effect of the intervention is modeled by fitting a single regression coefficient to the intervention's indicator variable.

When you set the number of lags greater than zero, regression coefficients are fit to lags of the indicator variable. This allows you to model interventions whose effects take place gradually, or to model rebound effects. For example, severe weather may reduce production during one period but cause an increase in production in the following period as producers struggle to catch up. You could model this using a point intervention with an effect time window of 1 lag. This would fit two coefficients for the intervention, one for the immediate effect and one for the delayed effect.

The `Effect Decay Pattern` option controls how the effect of the intervention dissipates over time. `None` specifies that there is no gradual decay: for point interventions, the effect ends immediately; for step and ramp interventions, the effect continues indefinitely. `Exp` specifies that the effect declines at an exponential rate. `Wave` specifies that the effect declines like an exponentially damped sine wave (or as the sum of two exponentials, depending on the fit to the data).

If you are familiar with time series analysis, these options may be clearer if you note that together the Effect Time Window and Effect Decay Pattern options define the numerator and denominator orders of a transfer function or dynamic regression model for the indicator variable of the intervention. See the section "Dynamic Regressor" later in this chapter for more information.

For this example, select 2 lags as the value of the Event Time Window option, and select `Exp` as the Effect Decay Pattern option. The window should now appear as shown in Figure 38.20.


SAS
File View Tools Solutions Window Help
Intervention Specification
OK
Cancel
Reset
Clear
Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products

Label: Point:AUG1990(2)/(1)

Intervention Specification:

Date: AUG1990

Type of Intervention: [✓ Point] Step Ramp

Effect Time Window:

Number of lags: 2

Effect Decay Pattern: None [✓ Exp] Wave

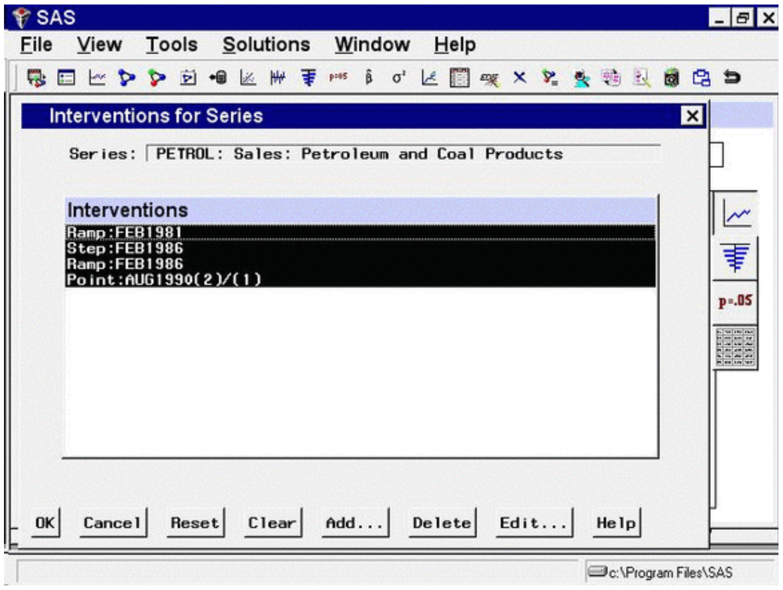
DATE	PETROL
MAY90	12995
JUN90	13039
JUL90	13035
AUG90	16683
SEP90	18752
OCT90	19604
NOV90	18201
DEC90	16080
JAN91	14935
FEB91	13261

Figure 38.20. Complex Intervention Model

Select the `OK` button to add the intervention to the list.

Fitting the Intervention Model

The Interventions for Series window now contains definitions for four intervention predictors. Select all four interventions, as shown in Figure 38.21.



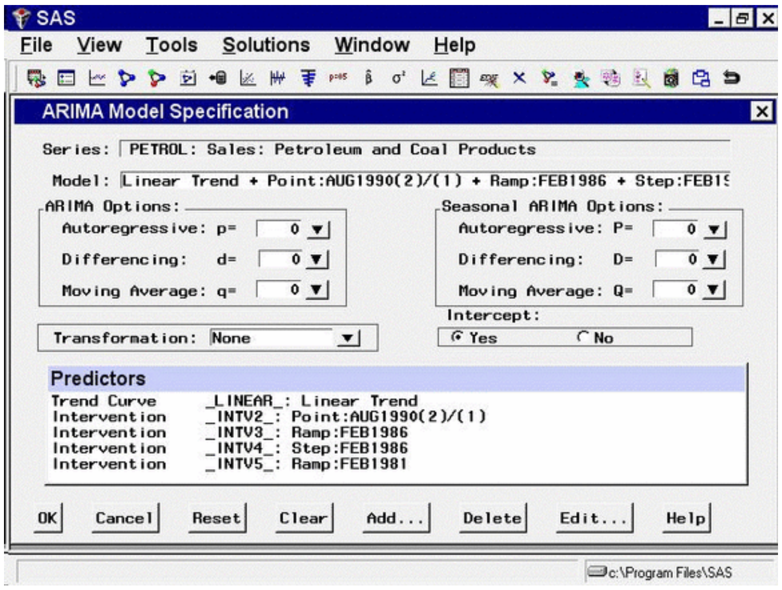
Series: PETROL: Sales: Petroleum and Coal Products

Interventions

Ramp:FEB1981
Step:FEB1986
Ramp:FEB1986
Point:AUG1990(2)/(1)

Figure 38.21. Interventions for Series Window

Select the `OK` button. This returns you to the ARIMA Model Specification window, which now lists items in the Predictors list, as shown in Figure 38.22.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products

Model: Linear Trend + Point:AUG1990(2)/(1) + Ramp:FEB1986 + Step:FEB1

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Transformation: None

Intercept: [✓ Yes] No

Predictors

Trend Curve	_LINEAR_: Linear Trend
Intervention	_INTV2_: Point:AUG1990(2)/(1)
Intervention	_INTV3_: Ramp:FEB1986
Intervention	_INTV4_: Step:FEB1986
Intervention	_INTV5_: Ramp:FEB1981

Figure 38.22. Linear Trend with Interventions Specified

Select the `OK` button to fit this model. After the model is fit, bring up the Model Viewer. You will see a plot of the model predictions, as shown in Figure 38.23.

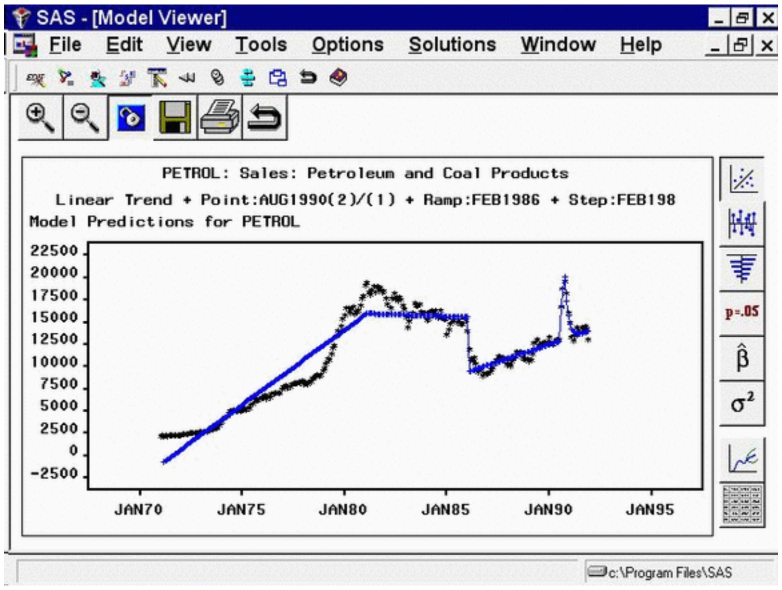

SAS - [Model Viewer]
File Edit View Tools Options Solutions Window Help
PETROL: Sales: Petroleum and Coal Products
Linear Trend + Point:AUG1990(2)/(1) + Ramp:FEB1986 + Step:FEB198
Model Predictions for PETROL
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
-2500
JAN70
JAN75
JAN80
JAN85
JAN90
JAN95
p=.05
c:\Program Files\SAS

Figure 38.23. Linear Trend with Interventions Model

You may wish to use the Zoom In feature to take a closer look at how the complex intervention effect fits the excursion in the series starting in August 1990.

Limitations of Intervention Predictors

Note that the model you have just fit is intended only to illustrate the specification of interventions. It is not intended as an example of good forecasting practice.

The use of continuing (step and ramp type) interventions as predictors has some limitations that you should consider. If you model a change in trend with a simple ramp intervention, then the trend in the data before the date of the intervention has no influence on the forecasts. Likewise, when you use a step intervention, the average level of the series before the intervention has no influence on the forecasts.

Only the final trend and level at the end of the series are extrapolated into the forecast period. If a linear trend is the only pattern of interest, then instead of specifying step or ramp interventions, it would be simpler to adjust the period of fit so that the model ignores the data before the final trend or level change.

Step and ramp interventions are valuable when there are other patterns in the data–such as seasonality, autocorrelated errors, and error variance–that are stable across the changes in level or trend. Step and ramp interventions allow you to fit seasonal and error autocorrelation patterns to the whole series while fitting the trend only to the latter part of the series.

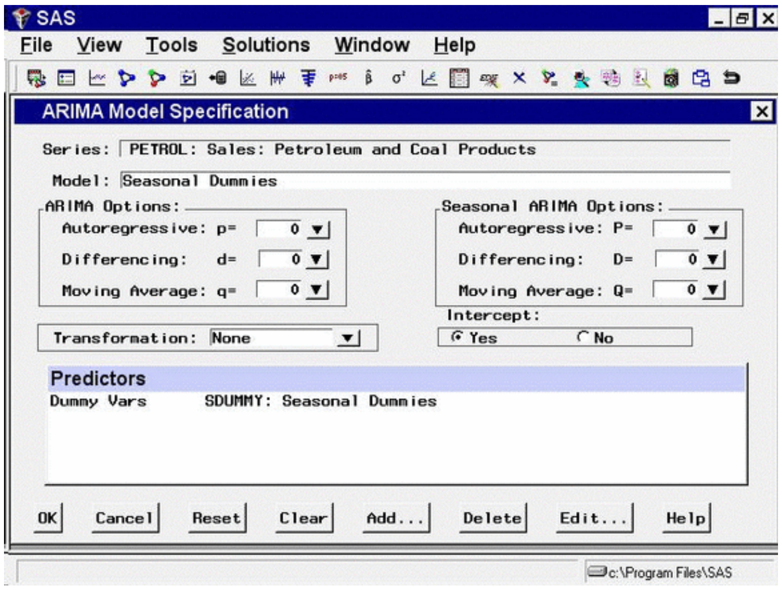
Point interventions are a useful tool for dealing with outliers in the data. A point intervention will fit the series value at the specified date exactly, and it has the effect of removing that point from the analysis. When you specify an effect time window, a point intervention will exactly fit as many additional points as the number of lags specified.

Seasonal Dummies

A *Seasonal Dummies* predictor is a special feature that adds to the model seasonal indicator or "dummy" variables to serve as regressors for seasonal effects.

From the Develop Models window, select `Fit ARIMA Model.` From the ARIMA Model Specification window, select `Add` and then select `Seasonal Dummies` from the menu (shown in Figure 38.1).

A Seasonal Dummies input is added to the Predictors list, as shown in Figure 38.24.


SAS
File View Tools Solutions Window Help
ARIMA Model Specification
OK Cancel Reset Clear Add... Delete Edit... Help
c:\Program Files\SAS

Series: PETROL: Sales: Petroleum and Coal Products
Model: Seasonal Dummies

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Transformation: None
Intercept: [✓ Yes] No

Predictors
Dummy Vars SDUMMY: Seasonal Dummies

Figure 38.24. Seasonal Dummies Specified

Select the OK button. A model consisting of an intercept and 11 seasonal dummy variables is fit and added to the model list in the Develop Models window. This is effectively a mean model with a separate mean for each month.

Now return to the Model Viewer, which displays a plot of the model predictions and actual series values, as shown in Figure 38.25. This is obviously a poor model for this series, but it serves to illustrate how seasonal dummy variables work.

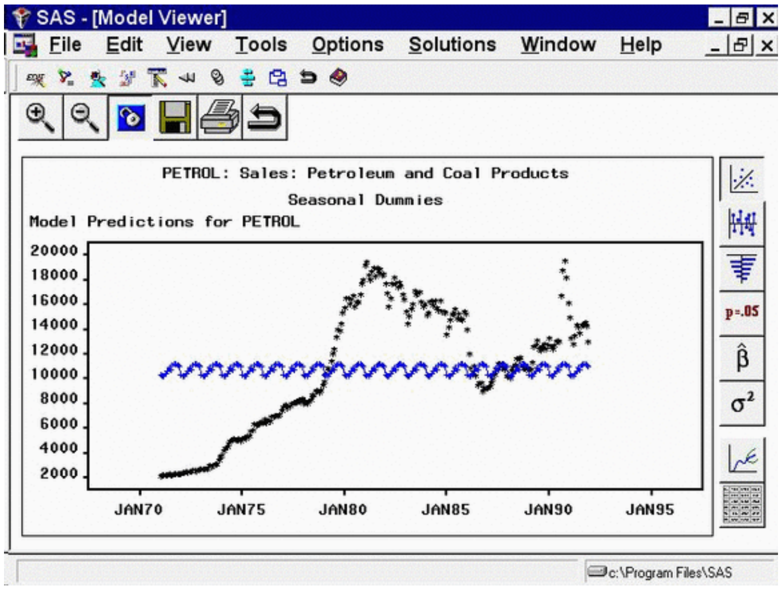
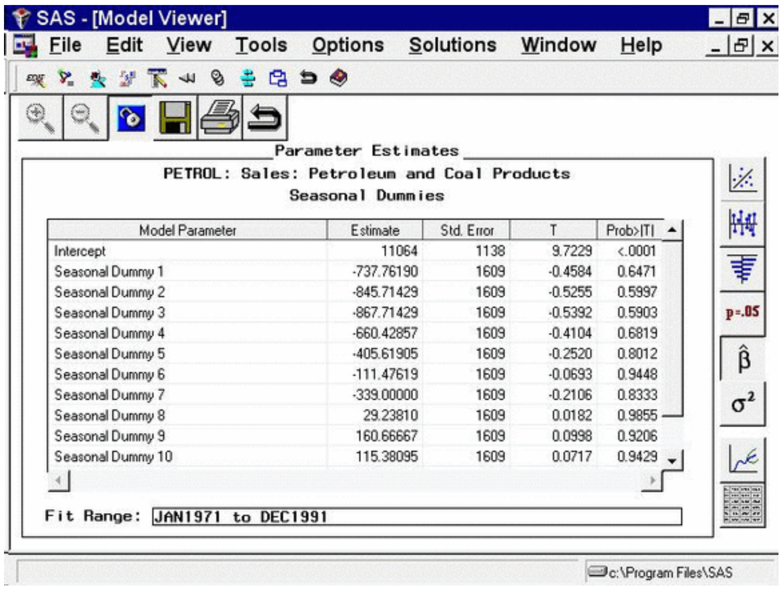


Figure 38.25. Seasonal Dummies Model

Now select the parameter estimates icon, the fifth from the top on the vertical toolbar. This displays the `Parameter Estimates` table, as shown in Figure 38.26.



Parameter Estimates

PETROL: Sales: Petroleum and Coal Products

Seasonal Dummies

Model Parameter	Estimate	Std. Error	T	Prob>|T|
Intercept	11064	1138	9.7229	<.0001
Seasonal Dummy 1	-737.76190	1609	-0.4584	0.6471
Seasonal Dummy 2	-845.71429	1609	-0.5255	0.5997
Seasonal Dummy 3	-867.71429	1609	-0.5392	0.5903
Seasonal Dummy 4	-660.42857	1609	-0.4104	0.6819
Seasonal Dummy 5	-405.61905	1609	-0.2520	0.8012
Seasonal Dummy 6	-111.47619	1609	-0.0693	0.9448
Seasonal Dummy 7	-339.00000	1609	-0.2106	0.8333
Seasonal Dummy 8	29.23810	1609	0.0182	0.9855
Seasonal Dummy 9	160.66667	1609	0.0998	0.9206
Seasonal Dummy 10	115.38095	1609	0.0717	0.9429

Fit Range: JAN1971 to DEC1991

Figure 38.26. Parameter Estimates for Seasonal Dummies Model

Since the data for this example are monthly, the Seasonal Dummies option added 11 seasonal dummy variables. These include a dummy regressor variable that is 1.0 for January and 0 for other months, a regressor that is 1.0 only for February, and so forth through November.

Because the model includes an intercept, no dummy variable is added for December. The December effect is measured by the intercept, while the effect of other seasons is measured by the difference between the intercept and the estimated regression coefficient for the season's dummy variable.

The same principle applies for other data frequencies: the "Seasonal Dummy 1" parameter will always refer to the first period in the seasonal cycle; and, when an intercept is present in the model, there will be no seasonal dummy parameter for the last period in the seasonal cycle.

References

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.

Pankratz, Alan (1991), *Forecasting with Dynamic Regression Models*, New York: John Wiley & Sons, Inc.

Chapter 39
Command Reference

Chapter Contents

Chapter 39
Command Reference

TSVIEW Command and Macro

The TSVIEW command invokes the Time Series Viewer. This is a component of the Time Series Forecasting System that can also be used as a stand-alone graphical viewer for any time series data set or view. See “Time Series Viewer Window” in Chapter 40, “Window Reference,” for more information.

The TSVIEW command must be given from the command line or an SCL program. If you need to submit from the program editor, use the %TSVIEW macro instead. You can use the macro within a data step program, but you must submit it within the SAS windowing environment.

If the TSVIEW command or %TSVIEW macro is issued without arguments, the Series Selection window appears to enable you to select an input data set and series. This is equivalent to selecting “Time Series Viewer” from the Analysis submenu of the Solutions pull-down menu. By specifying the DATA= and VAR= arguments, you can bring up the Time Series Viewer window directly. The ID= and INTERVAL= arguments are useful when the system cannot determine them automatically from the data.

Syntax

The TSVIEW command has the following form:

TSVIEW *[options]*

The %TSVIEW macro has the following form:

%TSVIEW *[(option, ..., option)] ;*

The following options can be specified for the command and the macro.

DATA= *data set name*
Specifies the name of the SAS data set containing the input data.

VAR= *time series variable name*
Specifies the series variable name. It must be a numeric variable contained in the data set.

ID= *time id variable name*
Specifies the time ID variable name for the data set. If the ID= option is not specified, the system will attempt to locate the variables named DATE, DATETIME, and TIME in the data set specified by the DATA= option.

INTERVAL= *interval name*
Specifies the time ID interval between observations in the data set.

Examples

TSVIEW Command

```
tsview data=sashelp.air var=air
tsview data=dept.prod var=units id=period interval=qtr
```

%TSVIEW Macro

```
%tsview( data=sashelp.air, var=air);
%tsview( data=dept.prod, var=units, id=period, interval=qtr);
```

FORECAST Command and Macro

The FORECAST command invokes the Time Series Forecasting System. The command must be given from the command line or an SCL program. If you need to submit from the program editor, use the %FORECAST macro instead. You can use the macro within a data step program, but you must submit it within the SAS windowing environment.

If the FORECAST command or %FORECAST macro is issued without arguments, the Time Series Forecasting window appears. This is equivalent to selecting "Time Series Forecasting System" from the `Analysis` submenu of the Solutions pulldown menu.

Using the arguments, it is possible to do the following:

- Bring up the system with information already filled into some of the fields
- Bring up the system starting at a different window than the default Time Series Forecasting window
- Run the system in unattended mode so that a task such as creating a forecast data set is accomplished without any user interaction. By submitting such commands repeatedly from a SAS/AF or SAS/EIS application, it is possible to do "batch" processing for many data sets or by-group processing for many subsets of a data set. You can create a project in unattended mode and later open it for inspection interactively. You can also create a project interactively in order to set options, fit a model, or edit the list of models, and then use this project later in unattended mode.

The `Forecast Command Builder`, a point-and-click SAS/AF application, makes it easy to specify, run, save, and rerun forecasting jobs using the FORECAST command. To use it, enter the following on the command line (not the program editor):

`%FCB`

or

`AF C=SASHELP.FORCAST.FORCCMD.FRAME`.

Syntax

The FORECAST command has the following form:

FORECAST *[options]*

The %FORECAST macro has the following form:

%FORECAST *[(option, ..., option)] ;*

The following options can be specified for the command and the macro.

PROJECT= *project name*
Specifies the name of the SAS catalog entry in which forecasting models and other results will be stored and from which previously stored results are loaded into the forecasting system.

DATA= *data set name*
Specifies the name of the SAS data set containing the input data.

VAR= *time series variable name*
Specifies the series variable name. It must be a numeric variable contained in the data set.

ID= *time id variable name*
Specifies the time ID variable name for the data set. If the ID= option is not specified, the system will attempt to locate the variables named DATE, DATETIME, and TIME in the data set specified by the DATA= option. However, it is recommended that you specify the time ID variable whenever you are using the ENTRY= argument.

INTERVAL= *interval name*
Specifies the time ID interval between observations in the data set. Commonly used intervals are `year`, `semiyear`, `qtr`, `month`, `semimonth`, `week`, `weekday`, `day`, `hour`, `minute`, and `second`. See Chapter 3, "Date Intervals, Formats, and Functions," for information on more complex interval specifications. If the INTERVAL= option is not specified, the system will attempt to determine the interval based on the time ID variable. However, it is recommended that you specify the interval whenever you are using the ENTRY= argument.

STAT= *statistic*
Specifies the name of the goodness-of-fit statistic to be used as the model selection criterion. The default is RMSE. Valid names are

`sse`	Sum of Square Error
`mse`	Mean Square Error
`rmse`	Root Mean Square Error
`mae`	Mean Absolute Error
`mape`	Mean Absolute Percent Error
`aic`	Akaike Information Criterion
`sbc`	Schwarz Bayesian Information Criterion

`rsquare`	R-Square
`adjrsq`	Adjusted R-Square
`rwrsq`	Random Walk R-Square
`arsq`	Amemiya's Adjusted R-Square
`apc`	Amemiya's Prediction Criterion

CLIMIT= *integer*
An integer specifying the level of the confidence limits to be computed for the forecast. This integer represents a percentage; for example, 925 indicates 92.5% confidence limits. The default is 95, that is, 95% confidence limits.

HORIZON= *integer*
Specifies the number of periods into the future for which forecasts will be computed. The default is 12 periods. The maximum is 9999.

ENTRY= *name*
The name of an entry point into the system. Valid names are

`main`	Starts the system at the Time Series Forecasting window (default).
`devmod`	Starts the system at the Develop Models window.
`viewmod`	Starts the system at the Model Viewer window. Specify a project containing a forecasting model using the PROJECT= option. If a project containing a model is not specified, the message "No forecasting model to view" appears.
`viewser`	Starts the system at the Time Series Viewer window.
`autofit`	Runs the system in unattended mode, fitting a forecasting model automatically and saving it in a project. If PROJECT= is not specified, the default project name SASUSER.FMSPROJ.PROJ is used.
`forecast`	Runs the system in unattended mode to generate a forecast data set. The name of this data set is specified by the OUT= parameter. If OUT= is not specified, a window appears to prompt for the name and label of the output data set. If PROJECT= is not specified, the default project name SASUSER.FMSPROJ.PROJ is used. If the project does not exist or does not contain a forecasting model for the specified series, automatic model fitting is performed and the forecast is computed using the automatically selected model. If the project exists and contains a forecasting model for the specified series, the forecast is computed using this model. If the series covers a different time range than it did when the project was created, use the REFIT or REEVAL keyword to reset the time ranges.

OUT= *argument*
The one or two-level name of a SAS data set in which forecasts will be saved. Use in conjunction with ENTRY=FORECAST. If omitted, the system prompts for the name of the forecast data set.

KEEP= *argument*

Specifies the number of models to keep in the project when automatic model fitting is performed. This corresponds to “Models to Keep” in the Automatic Model Selection Options window. A value greater than 9 indicates that all models will be kept. The default is 1.

DIAG= YES|NO

DIAG= YES causes the automatic model selection process to search only over those models that are consistent with the series diagnostics. DIAG= NO causes the automatic model selection process to search over all models in the selection list, without regard for the series diagnostics. This corresponds to `Models to Fit` in the Automatic Model Selection Options window. The default is YES.

REFIT *keyword*

Macro usage: REFIT= . Refits a previously saved forecasting model using the current fit range; the is, it re-estimates the model parameters. Refitting also causes the model to be reevaluated (statistics of fit recomputed), and it causes the time ranges to be reset if the data range has changed (for example, if new observations have been added to the series). This keyword has no effect if you do not use the PROJECT= argument to reference an existing project containing a forecasting model. Use the REFIT keyword if you have added new data to the input series and you want to refit the forecasting model and update the forecast using the new time ranges. Be sure to use the same project, data set, and series names that you used previously.

REEVAL *keyword*

Macro usage: REEVAL= . Re-evaluates a previously saved forecasting model using the current evaluation range; that is, it recomputes the statistics of fit. Re-evaluating also causes the time ranges to be reset if the data range has changed (for example, if new observations have been added to the series). It does not refit the model parameters. This keyword has no effect if you also specify REFIT, or if you do not use the PROJECT= argument to reference an existing project containing a forecasting model. Use the REEVAL keyword if you have added new data to the input series and want to update your forecast using a previously fit forecasting model and the same project, data set, and series names that you used previously.

Examples

FORECAST Command

The following command brings up the Time Series Forecasting window with the data set name and series name filled in. The time ID variable is also filled in since the data set contains the variable DATE. The interval is filled in because the system recognizes that the observations are monthly.

```
forecast data=sashelp.air var=air
```

The following command brings up the Time Series Forecasting window with the project, data set name, series, time ID, and interval fields filled in, assuming that the project SAMPROJ was previously saved either interactively or using unattended mode as depicted below. Previously fit models will appear when the Develop Models or Manage Projects window is opened.

```
forecast project=samproj
```

The following command runs the system in unattended mode, fitting a model automatically, storing it in the project SAMPROJ in the default catalog SASUSER.FMSPROJ, and placing the forecasts in the data set WORK.SAMPOUT.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=forecast out=sampout
```

The following command assumes that a new month's data have been added to the data set from the previous example and that an updated forecast is needed using the previously fit model. Time ranges are automatically updated to include the new data since the REEVAL keyword is included. Substitute REFIT for REEVAL if you want the system to re-estimate the model parameters.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=forecast out=sampout reeval
```

The following command brings up the model viewer using the project created in the previous example and using 99 percent confidence limits in the forecast graph.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=viewmod climit=99
```

The final example illustrates using unattended mode with an existing project that has been defined interactively. In this example, the goal is to add a model to the model selection list, and to specify that all models in that list be fit and that all models which are fit successfully be retained.

First bring up the Time Series Forecasting window and specify a new project name, WORKPROJ. Then select `Develop Models`, choosing SASHELP.WORKERS as the data set and MASONRY as the series. Now select "Model Selection List" from the Options pull-down menu. In the Model Selection List window, click `Actions`, then `Add`, and then `ARIMA Model`. Define the model `ARIMA(0,1,0)(0,1,0)s NOINT` by setting the differencing value to 1 under both `ARIMA Options` and `Seasonal ARIMA Options`. Select "OK" to save the model and `OK` to close the Model Selection List window. Now select "Automatic Fit" from the Options pull-down menu. In the Automatic Model Selection Options window, select "All autofit models in selection list" in the Models to fit radio box, select "All models" from the `Models to keep` combo box, and then click `OK` to close the window. Select "Save Project" from the File pull-down menu, and then close the Develop Models window and the Time Series Forecasting window. You now have a project with a new model added to the selection list, options set for automatic model fitting, and one series selected but no models fit.

Now enter the command:

```
forecast data=sashelp.workers var=electric id=date interval=month
project=workproj entry=forecast out=workforc
```

The system runs in unattended mode to update the project and create the forecast data set WORKFORC. Check the messages in the Log window to find out if the run was successful and which model was selected for forecasting. To see the forecast data set, issue the command `viewtable WORKFORC`. To see the contents of the project, bring up the Time Series Forecasting window, open the project WORKPROJ, and select “Manage Projects”. You will see that the variable ELECTRIC was added to the project and has a forecasting model. Select this row in the table and then select `List Models` from the Tools pull-down menu. You will see that all of the models in the selection list which fit successfully are there, including the new model you added to the selection list.

%FORECAST Macro

This example demonstrates the use of the %FORECAST macro to start the Time Series Forecasting System from a SAS program submitted from the Editor window. The SQL procedure is used to create a view of a subset of a products data set. Then the %FORECAST macro is used to produce forecasts.

```
proc sql;
create view selprod as
select * from products
where type eq ’A’
order by date;

%forecast(data=selprod, var=amount, id=date, interval=day,
     entry=forecast, out=typea, proj=proda, refit= );
```

Chapter 40
Window Reference

Chapter Contents

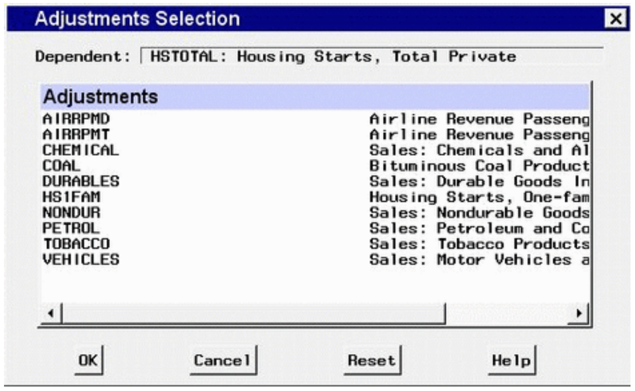
Chapter 40
Window Reference

Overview

This chapter provides a reference to the various windows of the Time Series Forecasting system. The windows are presented in alphabetical order by name. Each section describes the purpose of the window, how to bring it up, and its controls, fields, and menus. For windows which have their own pull-down menus, there is a description of each menu item under the heading “Menu Bar”. These windows also have a tool bar with icons duplicating the more commonly used menu items. Each icon has a *screen tip*: A brief description which appears when you place the mouse cursor over the icon. If you don’t see the screen tips, bring up the SAS Preferences window, under the Options submenu of the Tools pull-down menu. Select the View tab and make sure the “Screen tips” check box is checked.

Adjustments Selection Window

Use the Adjustments Selection window to select input variables for use as adjustments to the forecasts and add them to the Predictors list. Invoke this window from the pop-up menu which appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window. For more information, see the “Adjustments” section in Chapter 38, “Using Predictor Variables.”



Dependent: HSTOTAL: Housing Starts, Total Private

Adjustments	
AIRRPMD	Airline Revenue Passeng
AIRRPMT	Airline Revenue Passeng
CHEMICAL	Sales: Chemicals and Al
COAL	Bituminous Coal Product
DURABLES	Sales: Durable Goods In
HS1FAM	Housing Starts, One-fam
NONDUR	Sales: Nondurable Goods
PETROL	Sales: Petroleum and Co
TOBACCO	Sales: Tobacco Products
VEHICLES	Sales: Motor Vehicles a

Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Adjustments`
: is a table listing the names and labels in the input data set available for selection as adjustments. The variables you select are highlighted. Selecting a highlighted row again deselects that variable.

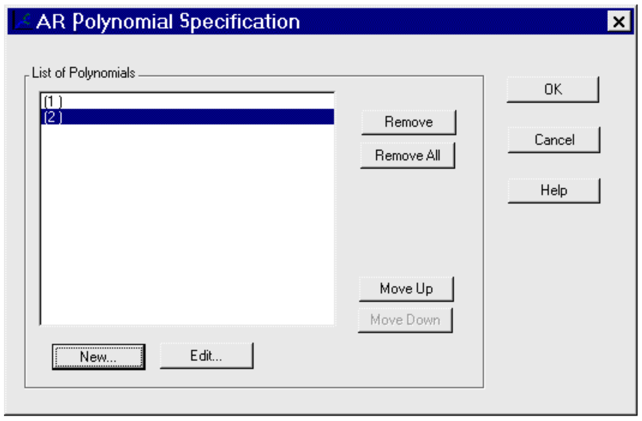
`OK`
: closes the Adjustments Selection window and adds the selected variables as adjustments in the model.

`Cancel`
: closes the window without adding any adjustments.

`Reset`
: resets all selections to their initial values upon entry to the window.

AR/MA Polynomial Specification Window

Use these windows to specify the autoregressive and moving average terms in a factored ARIMA model. Access the AR Polynomial Specification window from the Set button next to the Autoregressive term in the Factored ARIMA Model Specification window. Access the MA Polynomial Specification window from the Set button next to the Moving Average term.



Controls and Fields

`List of Polynomials`

Lists the polynomials which have been specified. Each polynomial is represented by a comma-delimited list of lag values enclosed in parentheses.

`New`
: Brings up the Polynomial Specification window to add a new polynomial to the model.

`Edit`
: Brings up the Polynomial Specification window to edit a polynomial which has been selected. If no polynomial is selected, this button is grayed.

`Remove`
: Removes a selected polynomial from the list. been selected. If none are selected, this button is grayed.

`Remove All`
: Clears the list of polynomials.

`Move Up`
: Moves a selected polynomial up one position in the list. If no polynomial is selected, or the first one is selected, this button is grayed.

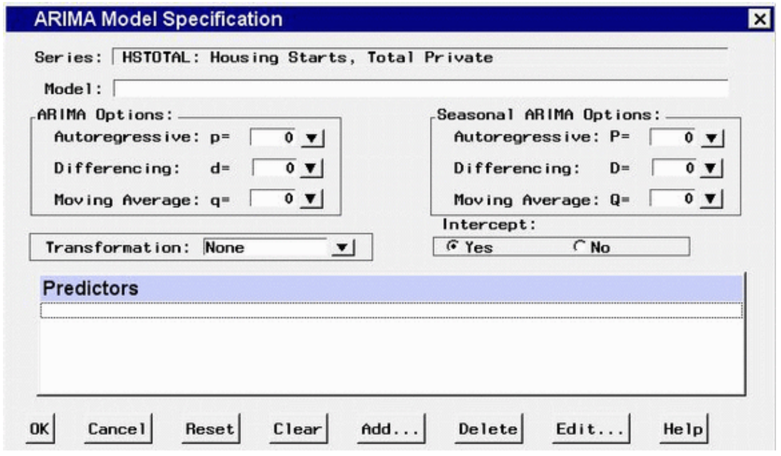
`Move Down`
: Moves a selected polynomial down one position in the list. If no polynomial is selected, or the last one is selected, this button is grayed.

`OK`
: Closes the window and returns the specified list of polynomials to the Factored ARIMA Model Specification window.

`Cancel`
: Closes the window and discards any changes made to the list of polynomials.

ARIMA Model Specification Window

Use the ARIMA Model Specification window to specify and fit an ARIMA model with or without predictor effects as inputs. Access it from the Develop Models menu, where it is invoked from the Fit Model item under Edit in the menu bar, or from the pop-up menu when you click an empty area of the model table.


ARIMA Model Specification

Series: HSTOTAL: Housing Starts, Total Private

Model:

ARIMA Options:
Autoregressive: p= 0
Differencing: d= 0
Moving Average: q= 0

Seasonal ARIMA Options:
Autoregressive: P= 0
Differencing: D= 0
Moving Average: Q= 0

Intercept: [✓ Yes] No

Transformation: None

Predictors

Controls and Fields

`Series`
: is the name and variable label of the current series.

`Model`
: is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`ARIMA Options`
: Use these combo boxes to specify the orders of the ARIMA model. You can either type in a value or click the combo box arrow to select from a popup list.

`Autoregressive`
defines the order of the autoregressive part of the model.

`Differencing`
defines the order of simple differencing, for example, first difference or second difference.

`Moving Average`
defines the order of the moving average part of the model.

`Seasonal ARIMA Options`
Use these combo boxes to specify the orders of the seasonal part of the ARIMA model. You can either type in a value or click the combo box arrow to select from a popup list.

`Autoregressive`
defines the order of the seasonal autoregressive part of the model.

`Differencing`
defines the order of seasonal differencing, for example, first difference or second difference at the seasonal lags.

`Moving Average`
defines the order of the seasonal moving average part of the model.

`Transformation`
defines the series transformation for the model. When a transformation is specified, the ARIMA model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the ARIMA model predictions. The available transformations are: `Log`, `Logistic`, `Square Root`, `Box-Cox`, and `None`.

`Intercept`
The options `Yes` and `No` enable you to specify whether a mean or intercept parameter is included in the ARIMA model. By default, the Intercept option is set to No when the model includes differencing and Yes when there is no differencing.

`Predictors`
This table lists the predictor effects included as inputs in the model.

`OK`
closes the ARIMA Model Specification window and fits the model.

`Cancel`
closes the ARIMA Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the ARIMA Model Specification window. This may be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
resets all options to their default values.

`Add`
brings up a menu of types of predictors to add to the Predictors list.

`Delete`
: deletes the selected (highlighted) entry from the Predictors list.

`Edit`
: edits the selected (highlighted) entry in the Predictors list.

Mouse Button Actions

You can select or deselect entries in the Predictors list by positioning the mouse cursor over the entry and clicking the left mouse button. The selected (highlighted) predictor effect is acted on by the Delete and Edit buttons. Double-clicking on a predictor in the list invokes an appropriate edit action for that predictor.

If you position the mouse cursor over the Predictors list and click the right mouse button, the system displays the following menu of actions encompassing the features of the Add, Delete, and Edit buttons.

`Add Linear Trend`
: adds a Linear Trend item to the Predictors list.

`Add Trend Curve`
: brings up a menu of different time trend curves and adds the curve you select to the Predictors list. Certain trend curve specifications also set the Transformation field.

`Add Regressors`
: brings up the Regressors Selection window to enable you to select other series in the input data set as regressors to predict the dependent series and add them to the Predictors list.

`Add Adjustments`
: brings up the Adjustments Selection window to enable you to select other series in the input data set for use as adjustments to the forecasts and add them to the Predictors list.

`Add Dynamic Regressor`
: brings up the Dynamic Regressor Selection window to enable you to select a series in the input data set as a predictor of the dependent series and also specify a transfer function model for the effect of the predictor series.

`Add Interventions`
: brings up the Interventions for Series window to enable you to define and select intervention effects and add them to the Predictors list.

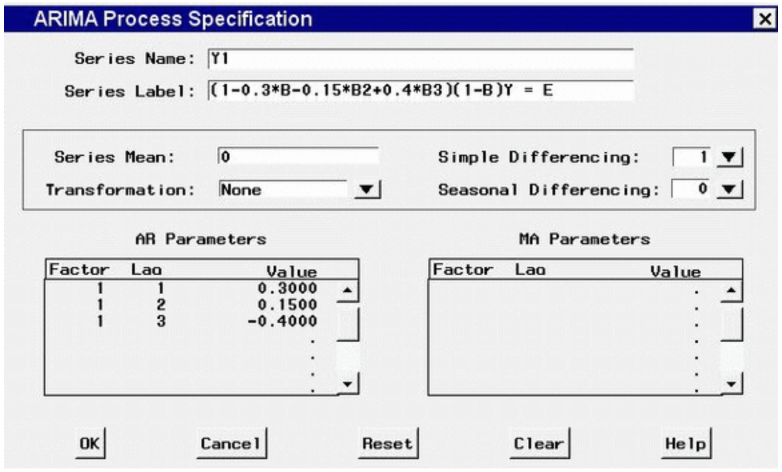
`Add Seasonal Dummies`
: adds a Seasonal Dummies predictor item to the Predictors list.

`Edit Predictor`
: edits the selected (highlighted) entry in the Predictors list.

`Delete Predictors`
: deletes the selected (highlighted) entry from the Predictors list.

ARIMA Process Specification Window

Use the ARIMA Process Specification window to define ARIMA processes for simulation. Invoke this window from the Add Series button in the Time Series Simulation window.



Series Name: Y1
Series Label: (1-0.3*B-0.15*B2+0.4*B3)(1-B)Y = E

Series Mean: 0
Transformation: None
Simple Differencing: 1
Seasonal Differencing: 0

AR Parameters

Factor	Lag	Value
1	1	0.3000
1	2	0.1500
1	3	-0.4000

MA Parameters

Factor	Lag	Value

Controls and Fields

`Series Name`
: is the variable name for the series to be simulated.

`Series Label`
: is the variable label for the series to be simulated.

`Series Mean`
: is the mean of the simulated series.

`Transformation`
: defines the series transformation.

`Simple Differencing`
: is the order of simple differencing for the series.

`Seasonal Differencing`
: is the order of seasonal differencing for the series.

`AR Parameters`
: is a table of Autoregressive terms for the simulated ARIMA process. Enter a value for Factor, Lag, and Value for each term of the AR part of the process you want to simulate. For a non-factored AR model, make the Factor values the same for all terms. For a factored AR model, use different Factor values to group the terms into the factors.

`MA Parameters`
: is a table of Moving Average terms for the simulated ARIMA process. Enter a value for Factor, Lag, and Value for each term of the MA part of the process you want to simulate. For a non-factored MA model, make the Factor values the same for all terms. For a factored MA model, use different Factor values to group the terms into the factors.

`OK`
: closes the ARIMA Process Specification window and adds the specified process to the Series to Generate list in the Time Series Simulation window.

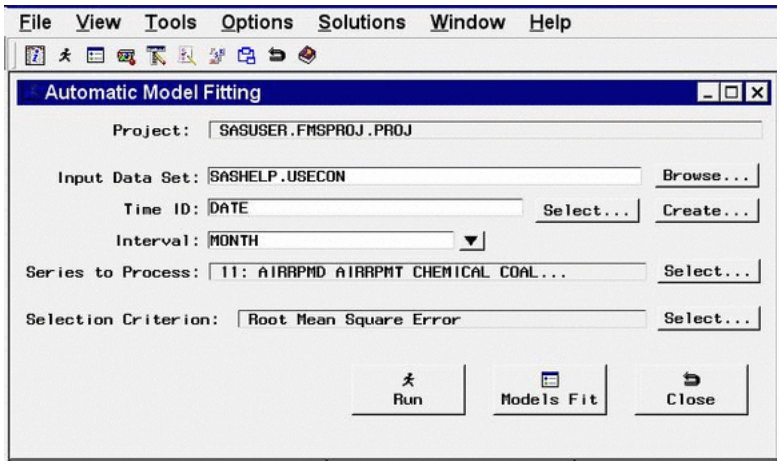
`Cancel`
: closes the window without adding to the Series to Generate list. Any options you specified are lost.

`Reset`
: resets all the fields to their initial values upon entry to the window.

`Clear`
: resets all the fields to their default values.

Automatic Model Fitting Window

Use the Automatic Model Fitting window to perform automatic model selection on all series or selected series in an input data set. Invoke this window using the Fit Models Automatically button on the Time Series Forecasting window. Note that you can also perform automatic model fitting, one series at a time, from the Develop Models window.



Project: SASUSER.FMSPROJ.PROJ

Input Data Set: SASHELP.USECON

Time ID: DATE

Interval: MONTH

Series to Process: 11: AIRRPMD AIRRPMT CHEMICAL COAL...

Selection Criterion: Root Mean Square Error

Controls and Fields

`Project`
: the name of the SAS catalog entry in which the results of the model search process are stored.

`Input Data Set`
: is the name of the current input data set. You can type in a one or two level data set name here.

`Browse button`
: brings up the Data Set Selection window for selecting an input data set.

`Time ID`
: is the name of the ID variable for the input data set. You can type in the variable name here, or use the Select or Create button.

time ID `Select` button
: brings up the Time ID Variable Specification window.

time ID `Create` button
: brings up a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the current input data set. You can type in an interval name or select one using the combo box popup menu.

`Series to Process`
: indicates the number and names of time series variables for which forecasting model selection will be applied.

Series to Process `Select` button
: brings up the Series to Process window to let you select the series for which you want to fit models.

`Selection Criterion`
: shows the goodness-of-fit statistic that will be used to determine the best fitting model for each series.

Selection Criterion `Select` button
: brings up the Model Selection Criterion window to enable you to select the goodness-of-fit statistic that will be used to determine the best fitting model for each series.

`Run` button
: begins the automatic model fitting process.

`Models Fit` button
: invokes the Automatic Model Fitting Results window to display the models fit during the current invocation of the Automatic Model Fitting window. The results appear automatically when model fitting is complete, but this button enables you to redisplay the results window.

`Close` button
: Closes the Automatic Model Fitting window.

Menu Bar

`File`

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Close`
: Closes the Automatic Model Fitting window.

`View`

`Input Data Set`
: brings up a Viewtable window to browse the current input data set.

`Models Fit`
: brings up Automatic Model Fitting Results window to show the forecasting models fit during the current invocation of the Automatic Model Fitting window. This is the same as the Models Fit button.

`Tools`

`Fit Models`
: performs the automatic model selection process for the selected series. This is the same as the Run button.

`Options`

`Default Time Ranges`
: brings up the Default Time Ranges window to enable you to control how the system sets the time ranges for series.

`Model Selection List`
: Brings up the Model Selection List editor window. Use this action to control the forecasting models considered by the automatic model selection process and displayed in the Models to Fit window.

`Model Selection Criterion`
: brings up the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model. This action is the same as the Selection Criterion Select button.

`Statistics of Fit`
: brings up the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Statistics of Fit table and available for selection in the Model Selection Criterion menu.

`Forecast Options`
: brings up the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Forecast Data Set`
: see Produce Forecasts window.

`Alignment of Dates`

`Beginning`
: aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

`Middle`
: aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

`End`
: aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
: brings up the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Tool Bar Type`

`Image Only`
: displays the tool bar items as icons without text.

`Label Only`
: displays the tool bar items as text without icon images.

`Both`
: displays the tool bar items with both text and icon images.

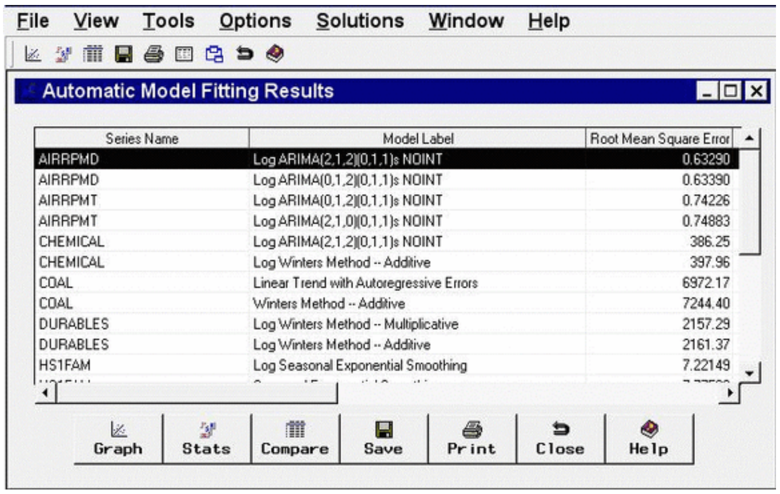
`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Automatic Model Fitting Results Window

This resizable window displays the models fit by the most recent invocation of the Automatic Model Fitting window. It appears automatically after Automatic Model Fitting runs, and can be brought up repeatedly from that window using the Models Fit button or by selecting Models Fit from the View pull-down menu. Once you exit the Automatic Model Fitting window, the Automatic Model Fitting Results window cannot be opened again until you fit additional models using Automatic Model Fitting.


File View Tools Options Solutions Window Help
Automatic Model Fitting Results
Graph
Stats
Compare
Save
Print
Close
Help

Series Name	Model Label	Root Mean Square Error
AIRRPMD	Log ARIMA(2,1,2)(0,1,1)s NOINT	0.63290
AIRRPMD	Log ARIMA(0,1,2)(0,1,1)s NOINT	0.63390
AIRRPMT	Log ARIMA(0,1,2)(0,1,1)s NOINT	0.74226
AIRRPMT	Log ARIMA(2,1,0)(0,1,1)s NOINT	0.74883
CHEMICAL	Log ARIMA(2,1,2)(0,1,1)s NOINT	386.25
CHEMICAL	Log Winters Method -- Additive	397.96
COAL	Linear Trend with Autoregressive Errors	6972.17
COAL	Winters Method -- Additive	7244.40
DURABLES	Log Winters Method -- Multiplicative	2157.29
DURABLES	Log Winters Method -- Additive	2161.37
HS1FAM	Log Seasonal Exponential Smoothing	7.22149

Table Contents

The results table displays the series name in the first column and the model label in the second column. If you have chosen to retain more than one model using the Automatic Model Selection Options window, more than one row appears in the table for each series; that is, there is a row for each model fit. If you have already fit models to the same series before invoking the Automatic Model Fitting window, those models do not appear here, since the Automatic Model Fitting Results window is intended to show the results of the current operation of Automatic Model Fitting. To see all models which have been fit, use the Manage Projects window.

The third column of the table shows the values of the current model selection criterion statistic. Additional columns show the values of other fit statistics. The set of statistics shown are selectable using the Statistics of Fit Selection window.

The table can be sorted by any column other than Series Name by clicking on the column heading.

Controls and Fields

`Graph`
: brings up the Model Viewer window on the model currently selected in the table.

`Stats`
: brings up the Statistics of Fit Selection window. This controls the set of

goodness-of-fit statistics displayed in the table and in other parts of the Time Series Forecasting System.

`Compare`
: brings up the Model Fit Comparison window for the series currently selected in the table. This button is grayed out if the currently selected row in the table represents a series for which fewer than two models have been fit.

`Save`
: brings up an output data set dialog, enabling you to specify a SAS data set to which the contents of the table will be saved. Note that this operation saves what you see in the table. If you want to save the models themselves for use in a future session, use the Manage Projects window.

`Print`
: prints the contents of the table.

`Close`
: closes the window and returns to the Automatic Model Fitting window.

Menu Bar

`File`

`Save`
: brings up an output data set dialog, enabling you to specify a SAS data set to which the contents of the table will be saved. This is the same as the Save button.

`Print`
: prints the contents of the table. This is the same as the Print button.

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the window and returns to the Automatic Model Fitting window.

`View`

`Model Predictions`
: brings up the Model Viewer to display a predicted and actual plot for the currently highlighted model.

`Prediction Errors`
: brings up the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
: brings up the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
: brings up the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
: brings up the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
: brings up the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
: brings up the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
: brings up the Model Viewer to display forecasts for the currently highlighted model in a table.

`Tools`

`Compare Models`
: brings up the Model Fit Comparison window to display fit statistics for selected pairs of forecasting models. This item is grayed out until you select a series in the table for which the automatic model fitting run selected two or more models.

`Options`

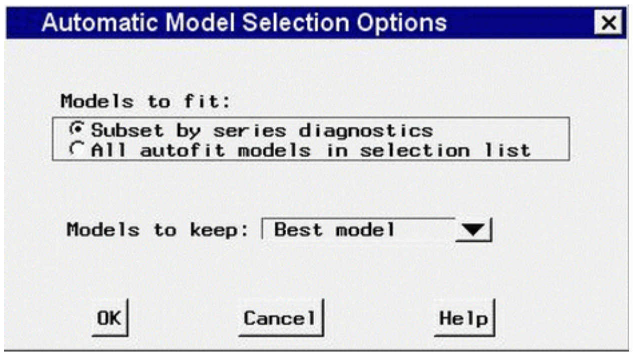
`Statistics of Fit`
: brings up the Statistics of Fit Selection window. This is the same as the Stats button.

`Column Labels`
: selects long or short column labels for the table. Long column labels are used by default.

`ID Columns`
: freezes or unfreezes the series and model columns. By default they are frozen so that they remain visible when you scroll the table horizontally to view other columns.

Automatic Model Selection Options Window

Use the Automatic Model Selection Options window to control the automatic selection process. This window is available from the Automatic Fit item of the Options pull-down menu in the Develop Models window, Automatic Model Fitting window, and Produce Forecasts window.


Automatic Model Selection Options

Models to fit:
[✓ Subset by series diagnostics]
All autofit models in selection list

Models to keep: Best model

Controls and Fields

`Models to fit`

`Subset by series diagnostics`
when selected, causes the automatic model selection process to search only over those models consistent with the series diagnostics.

`All models in selection list`
when selected, causes the automatic model selection process to search over all models in the search list, without regard for the series diagnostics.

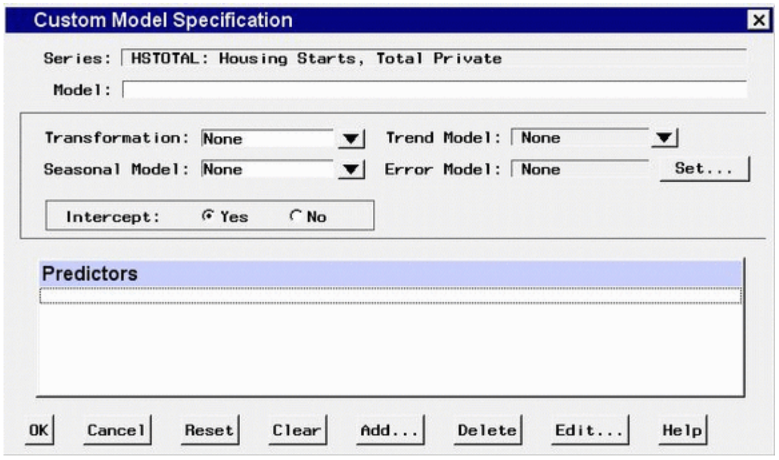
`Models to keep`
specifies how many of the models tried by the automatic model selection process are retained and added to the model list for the series. You can specify the best fitting model only, the best *n* models, where *n* can be 1 through 9, or all models tried.

`OK`
closes the window and saves the automatic model selection options you specified.

`Cancel`
closes the window without changing the automatic model selection options.

Custom Model Specification Window

Use the Custom Model Specification window to specify and fit an ARIMA model with or without predictor effects as inputs. Access it from the Develop Models window, where it is invoked from the Fit Model item under the Edit pull-down menu, or from the pop-up menu when you click an empty area of the model table.



Controls and Fields

`Series`
: is the name and variable label of the current series.

`Model`
: is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Transformation`
: defines the series transformation for the model. When a transformation is specified, the model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the resulting forecasts. The available transformations are

  `Log`
  : specifies a logarithmic transformation.

  `Logistic`
  : specifies a logistic transformation.

  `Square Root`
  : specifies a square root transformation.

  `Box-Cox`
  : specifies a Box-Cox transform and brings up a window to specify the Box-Cox λ parameter.

  `None`
  : specifies no series transformation.

`Trend Model`
: controls the model options to model and forecast the series trend. Select from

  `Linear Trend`
  : adds a Linear Trend item to the Predictors list.

`Trend Curve`
: brings of a menu of different time trend curves and adds the curve you select to the Predictors list.

`First Difference`
: specifies differencing the series.

`Second Difference`
: specifies second-order differencing of the series.

`None`
: specifies no model for the series trend.

`Seasonal Model`
: controls the model options to model and forecast the series seasonality. Select from:

`Seasonal ARIMA`
: brings up the Seasonal ARIMA Model Options window to enable you to specify an ARIMA model for the seasonal pattern in the series.

`Seasonal Difference`
: specifies differencing the series at the seasonal lag.

`Seasonal Dummy Regressors`
: adds a Seasonal Dummies predictor item to the Predictors list.

`None`
: specifies no seasonal model.

`Error Model`
: displays the current settings of the autoregressive and moving average terms, if any, for modeling the prediction error autocorrelation pattern in the series.

`Set` button
: brings up the Error Model Options window to let you set the autoregressive and moving average terms for modeling the prediction error autocorrelation pattern in the series.

`Intercept`
: The options “Yes” and “No” enable you to specify whether a mean or intercept parameter is included in the model. By default, the Intercept option is set to No when the model includes differencing and set to Yes when there is no differencing.

`Predictors`
: is a list of the predictor effects included as inputs in the model.

`OK`
: closes the Custom Model Specification window and fits the model.

`Cancel`
: closes the Custom Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the Custom Model

Specification window. This may be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

`Add`
: brings up a menu of types of predictors to add to the Predictors list. Select from:

  `Linear Trend`
  : adds a Linear Trend item to the Predictors list.

  `Trend Curve`
  : brings up a menu of different time trend curves and adds the curve you select to the Predictors list.

  `Regressors`
  : brings up the Regressors Selection window to enable you to select other series in the input data set as regressors to predict the dependent series and add them to the Predictors list.

  `Adjustments`
  : brings up the Adjustments Selection window to enable you to select other series in the input data set for use as adjustments to the forecasts and add them to the Predictors list.

  `Dynamic Regressor`
  : brings up the Dynamic Regressor Selection window to enable you to select a series in the input data set as a predictor of the dependent series and also specify a transfer function model for the effect of the predictor series.

  `Interventions`
  : brings up the Interventions for Series window to enable you to define and select intervention effects and add them to the Predictors list.

  `Seasonal Dummies`
  : adds a Seasonal Dummies predictor item to the Predictors list. This is grayed out if the series interval is not one which has a seasonal cycle.

`Delete`
: deletes the selected (highlighted) entry from the Predictors list.

`Edit`
: edits the selected (highlighted) entry in the Predictors list.

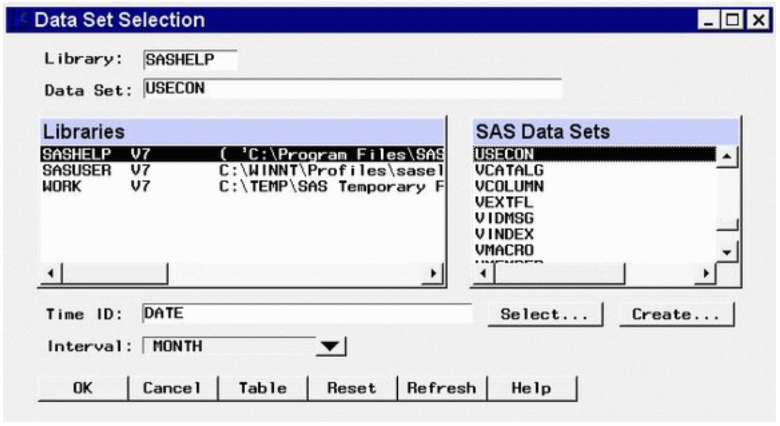
Mouse Button Actions

You can select or deselect entries in the Predictors list by positioning the mouse cursor over the entry and clicking the left mouse button. The selected (highlighted) predictor effect is acted on by the Delete and Edit buttons. Double-clicking on a predictor in the list invokes an appropriate edit action for that predictor.

If you position the mouse cursor over the Predictors list and press the right mouse button, the system displays a menu of actions encompassing the features of the Add, Delete, and Edit buttons.

Data Set Selection Window

Use this resizable window to select a data set to process by specifying a library and a SAS data set or view. These selections can be made by typing, by selecting from lists, or by a combination of the two. In addition, you can control the time ID variable and time interval, and you can browse the data set.



Library: SASHELP
Data Set: USECON

```
SASHELP   V7       ( 'C:\Program Files\SAS
SASUSER   V7       C:\WINNT\Profiles\sasel
WORK      V7       C:\TEMP\SAS Temporary F
```

```
USECON
VCATALG
VCOLUMN
VEXTFL
VIDMSG
VINDEX
VMACRO
```

Time ID: DATE
Interval: MONTH

Access this window using the Browse button to the right of the Data Set field in the Time Series Forecasting, Automatic Model Fitting, and Produce Forecasts windows. It functions in the same way as the Series Selection window, except that it does not allow you to select or view a time series variable.

Controls and Fields

`Library`
: is a SAS libname assigned within the current SAS session. If you know the libname associated with the data set of interest, you can type it in this field. If it is a valid choice, it will appear in the libraries list and will be highlighted. The SAS Data Sets list will be populated with data sets associated with that libname. See also Libraries under Selection Lists.

`Data Set`
: is the name of a SAS data set (data file or data view) that resides under the selected libname. If you know the name, you can type it in and press Return. If it is a valid choice, it will appear in the SAS Data Sets list and will be highlighted.

`Time ID`
: is the name of the ID variable for the selected input data set. To specify the ID variable, you can type the ID variable name in this field or select the control arrows to the right of the field.

Time ID `Select` button
: brings up the Time ID Variable Specification window.

Time ID `Create` button
: brings up a menu of methods for creating a time ID variable for the input data set. Use this feature if the data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the selected data set. If the interval is not automatically identified by the system, you can type in the interval name or select it from a list by clicking the combo box arrow. For more information on intervals, see Chapter 3, "Date Intervals, Formats, and Functions," in this book.

`OK`
: closes the Data Set Selection window and makes the selected data set the current input data set.

`Cancel`
: closes the window without applying any selections made.

`Table`
: brings up a Viewtable window for browsing the selected data set.

`Reset`
: resets the fields to their initial values upon entry to the window.

`Refresh`
: updates all fields and lists on the window. If you assign a new libname without exiting the Data Set Selection window, use the refresh action to update the Libraries list so that it will include the newly assigned libname.

Selection Lists

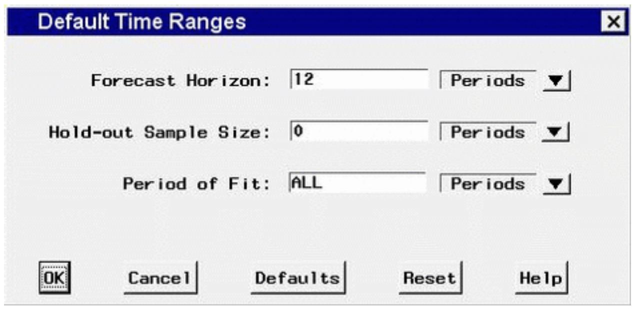
`Libraries`
: displays a list of currently assigned libnames. You can select a libname by clicking it with the left mouse button, which is equivalent to typing its name in the Library field.

  If you cannot locate the library or directory you are interested in, go to the SAS Explorer window, select "New" from the File pull-down menu, then select "Library" and "OK". This brings up the New Library window. You also assign a libname by submitting a libname statement from the Editor window. Select the Refresh button to make the new libname available in the libraries list.

`SAS Data Sets`
: displays a list of the SAS data sets (data files or data views) contained in the selected library. You can select one of these by clicking with the left mouse button, which is equivalent to typing its name in the Data set field. You can double-click on a data set name to select it and exit the window.

Default Time Ranges Window

Use the Default Time Ranges window to control how the period of fit and evaluation and the forecasting horizon are determined for each series when you do not explicitly set these ranges for a particular series. Invoke this window from the Options pull-down menu of the Develop Models, Automatic Model Fitting, Produce Forecasts, and Manage Forecasting Project windows. The settings you make in this window affect subsequently selected series: They do not alter the time ranges of series you have already selected.


Default Time Ranges

Forecast Horizon: 12 Periods

Hold-out Sample Size: 0 Periods

Period of Fit: ALL Periods

Controls and Fields

`Forecast Horizon`
: specifies the forecast horizon as either a number of periods or years from the last nonmissing data value or as a fixed date. You can type a number or date value in this field. Date value must be entered in a form recognized by a SAS date informat. (Refer to "SAS Language Reference" for information on SAS date informats.)

`Forecast Horizon Units`
: indicates whether the value in the forecast horizon field represents periods or years or a date. Click the arrow and choose one of these three options from the popup list.

`Hold-out Sample Size`
: specifies that a number of observations, number of years, or percent of the data at the end of the data range be used for the period of evaluation with the remainder of data used as the period of fit.

`Hold-out Sample Size Units`
: indicates whether the hold-out sample size represents periods or years or percent of data range.

`Period of Fit`
: specifies how much of the data range for a series is to be used as the period of fit for models fit to the series. ALL indicates that all the available data is used. You can specify a number of periods, number of years, or a fixed date, depending on the value of the units field to the right. When you specify a date, the start of the period of fit is the specified date or the first nonmissing series value, whichever is more recent. Date value must be entered in a form recognized by a SAS date informat. (Refer to "SAS Language Reference" for information on SAS date informats.) When you specify the number of periods or years, the start of the period of fit is computed as the date that number of periods or years from the end of the data.

`Period of Fit Units`
: indicates whether the period of fit value represents periods or years or a date.

`OK`
: closes the window and stores the specified changes.

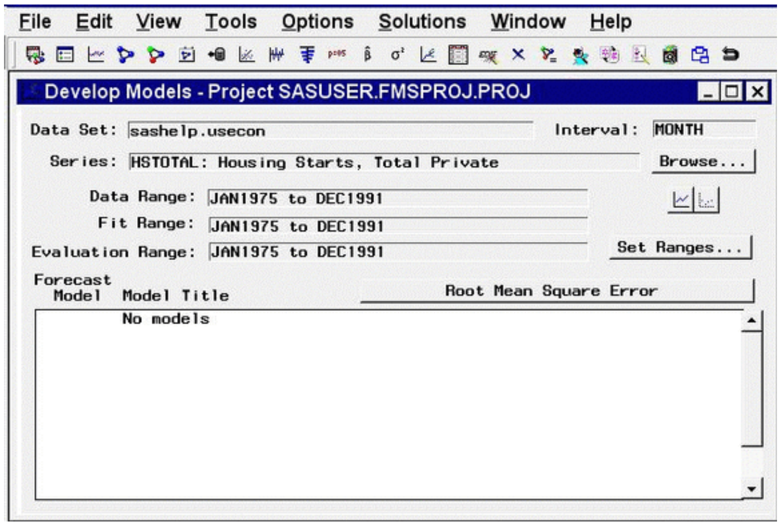
`Cancel`
: closes the window without saving changes. Any options you specified are lost.

`Defaults`
: resets all options to their default values.

`Reset`
: resets the options to their initial values upon entry to the window.

Develop Models Window

This resizable window provides access to all of the Forecasting System's interactive model fitting and graphical tools. Use it to fit forecasting models to an individual time series and choose the best model to use to produce the final forecasts of the series. Invoke this window using the Develop Models button on the Time Series Forecasting window.


File Edit View Tools Options Solutions Window Help
Develop Models - Project SASUSER.FMSPROJ.PROJ
Browse...
Set Ranges...

Data Set: sashelp.usecon
Interval: MONTH
Series: HSTOTAL: Housing Starts, Total Private
Data Range: JAN1975 to DEC1991
Fit Range: JAN1975 to DEC1991
Evaluation Range: JAN1975 to DEC1991

Forecast Model	Model Title	Root Mean Square Error
	No models	

Controls and Fields

`Data Set`
: is the name of the current input data set.

`Interval`
: is the time interval (data frequency) for the input data set.

`Series`
: is the variable name and label of the current time series.

`Browse` button
: brings up the Series Selection window to enable you to change the current input data set or series.

`Data Range`
: is the date of the first and last nonmissing data values available for the current series in the input data set.

`Fit Range`
: is the current period of fit setting. This is the range of data that will be used to fit models to the series.

`Evaluation Range`
: is the current period of evaluation setting. This is the range of data that will be used to calculate the goodness-of-fit statistics for models fit to the series.

`Set Ranges` button
: brings up the Time Ranges Specification window to enable you to change the fit range or evaluation range. Note: A new fit range is applied when new models are fit or when existing models are refit. A new evaluation range is applied when new models are fit or when existing models are refit or reevaluated. Changing the ranges does not automatically refit or reevaluate any models in the table: Use the Refit Models or Reevaluate Models items under the Edit pull-down menu.

`View Series Graphically` icon
: brings up the Time Series Viewer window to display plots of the current series.

`View Selected Model Graphically` icon
: brings up the Model Viewer to display graphs and tables for the currently highlighted model.

`Forecast Model`
: is the column of the model table that contains check boxes to select which model is used to produce the final forecasts for the current series.

`Model Title`
: is the column of the model table that contains the descriptive labels of the forecasting models fit to the current series.

`Root Mean Square Error` (or other statistic name) button
: is the button above the right side of the table. It displays the name of the current model selection criterion: A statistic that measures how well each model in the table fits the values of the current series for observations within the evaluation range. Clicking this button brings up the Model Selection Criterion window to let you to select a different statistic. When you select a statistic, the model table the Develop Models window is updated to show current values of that statistic.

Menu Bar

`File`

`New Project`
: brings up a dialog which lets you create a new project, assign it a name and description, and make it the active project.

`Open Project`
: brings up a dialog which lets you select and load a previously saved project.

`Save Project`
: saves the current state of the system (including all the models fit to a series) to the current project catalog entry.

`Save Project as`
: saves the current state of the system with a prompt for the name of the catalog entry in which to store the information.

`Clear Project`
: clears the system, deleting all the models for all series.

`Save Forecast`
: writes forecasts from the currently highlighted model to an output data set.

`Save Forecast As`
: prompts for an output data set name and saves the forecasts from the currently highlighted model.

`Output Forecast Data Set`
: brings up a dialog for specifying the default data set used when you select "Save Forecast".

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the Develop Models window and returns to the main window.

`Edit`

`Fit Model`

`Automatic Fit`
: invokes the automatic model selection process.

`Select From List`
: brings up the Models to Fit window.

`Smoothing Model`
: brings up the Smoothing Model Specification window.

`ARIMA Model`
: brings up the ARIMA Model Specification window.

`Custom Model`
: brings up the Custom Model Specification window.

`Combine Forecasts`
: brings up the Forecast Combination Model Specification window.

`External Forecasts`
: brings up the External Forecast Model Specification window.

`Edit Model`
: enables you to modify the specification of the currently highlighted model in the table and fit the modified model. The new model replaces the current model in the table.

`Delete Model`
: deletes the currently highlighted model from the model table.

`Refit Models`

`All Models`
: refits all models in the table using data within the current fit range.

`Selected Model`
: refits the currently highlighted model using data within the current fit range.

`Reevaluate Models`

`All Models`
: recomputes statistics of fit for all models in the table using data within the current evaluation range.

`Selected Model`
: recomputes statistics of fit for the currently highlighted model using data within the current evaluation range.

`View`

`Project`
: brings up the Manage Forecasting Project window.

`Data Set`
: brings up a Viewtable window to display the current input data set.

`Series`
: brings up the Time Series Viewer window to display plots of the current series. This is the same as the View Series Graphically icon.

`Model Predictions`
: brings up the Model Viewer to display a predicted versus actual plot for the currently highlighted model. This is the same as the View Selected Model Graphically icon.

`Prediction Errors`
: brings up the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
: brings up the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
: brings up the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
: brings up the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
: brings up the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
: brings up the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
: brings up the Model Viewer to display forecasts for the currently highlighted model in a table.

`Tools`

`Diagnose Series`
: brings up the Series Diagnostics window to determine the kinds of forecasting models appropriate for the current series.

`Define Interventions`
: brings up the Interventions for Series window to enable you to edit or add intervention effects for use in modeling the current series.

`Sort Models`
: sorts the models in the table by the values of the currently displayed fit statistic.

`Compare Models`
: brings up the Model Fit Comparison window to display fit statistics for selected pairs of forecasting models. This is grayed out if there are fewer than two models in the table.

`Generate Data`
: brings up the Time Series Simulation window. This window enables you to simulate ARIMA time series processes and is useful for educational exercises or testing the system.

`Options`

`Time Ranges`
: brings up the Time Ranges Specification window to enable you to change the fit and evaluation time ranges and the forecast horizon. This action is the same as the Set Ranges button.

`Default Time Ranges`
: brings up the Default Time Ranges window to enable you to control how the system sets the time ranges for series when you do not explicitly set time ranges with the Time Ranges Specification window. Settings made using this window do not affect series you are already working with, they take effect when you select a new series.

`Model Selection List`
: brings up the Model Selection List editor window. Use this action to edit

the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

`Model Selection Criterion`
brings up the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model. This action is the same as clicking the button above the table which displays the name of the current model selection criterion.

`Statistics of Fit`
brings up the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Model Viewer, Automatic Model Fitting Results, and Model Fit Comparison windows and available for selection in the Model Selection Criterion menu.

`Forecast Options`
brings up the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Alignment of Dates`

`Beginning`
aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

`Middle`
aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

`End`
aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
brings up the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Include Interventions`
controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process and displayed by the Models to Fit window. When the Include Interventions option is selected, the series interventions are also automatically added to the predictors list when you specify a model in the ARIMA and Custom Models Specification windows. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: Controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Left Mouse Button Actions for the Model Table

When the cursor is over the description of a model in the table, the left mouse button selects (highlights) or deselects that model. On some computer systems, you can double-click to bring up the Model Viewer window for the selected model.

When the cursor is over an empty part of the model table, the left mouse button brings up a menu of model fitting choices. These choices are the same as those in the Fit Model submenu of the of the Edit pull-down menu.

Right Mouse Button Actions for the Model Table

When a model in the table is selected, the right mouse brings up a menu of actions that apply to the highlighted model. The actions available in this menu are as follows.

`View Model`
: brings up the Model Viewer for the selected model. This action is the same as the View Model Graphically icon.

`View Parameter Estimates`
: brings up the Model Viewer to display the parameter estimates table for the currently highlighted model. This is the same as the Parameter Estimates item in the View pull-down menu.

`View Statistics of Fit`
: brings up the Model Viewer to display a table of goodness-of-fit statistics for the currently highlighted model. This is the same as the Statistics of Fit item in the View pull-down menu.

`Edit Model`
: enables you to modify the specification of the currently highlighted model in the table and fit the modified model. This is the same as the Edit Model item in the Edit pull-down menu.

`Refit Model`
: refits the highlighted model using data within the current fit range. This is the same as the Selected Model item under the Refit Models submenu of the Edit pull-down.

`Reevaluate Model`
: reevaluates the highlighted model using data within the evaluation fit range. This is the same as the Selected Model item under the Reevaluate Models submenu of the Edit pull-down.

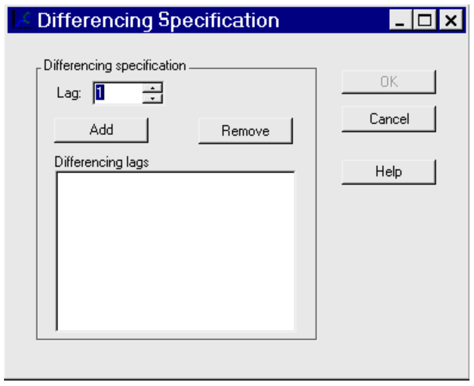
`Delete Model`
: deletes the currently highlighted model from the model table. This is the same as the Delete Model item under the Edit pull-down menu.

`View Forecasts`
: brings up the Model Viewer to display the forecasts for the currently highlighted model. This is the same as the Forecast Graph item under the View pull-down menu.

When the model list is empty or when no model is selected, the right mouse button brings up the same menu of model fitting actions as the left mouse button.

Differencing Specification Window

Use the Differencing Specification window to specify the list of differencing lags `d=(lag, ..., lag)` in a factored ARIMA model. To specify a first difference, add the value 1 (`d=(1)`. To specify a second difference (difference twice at lag 1), add the value 1 again `d=(1,1)`. For first differencing at lags 1 and 12, use the values 1 and 12 (`d=(1,12)`.



Controls and Fields

`Lag`
: specifies a lag value to add to the list. Type in a positive integer or select one by clicking the spin box arrows. Duplicates are allowed.

`Add`
: Adds the value in the `Lag` spin box to the list of differencing lags.

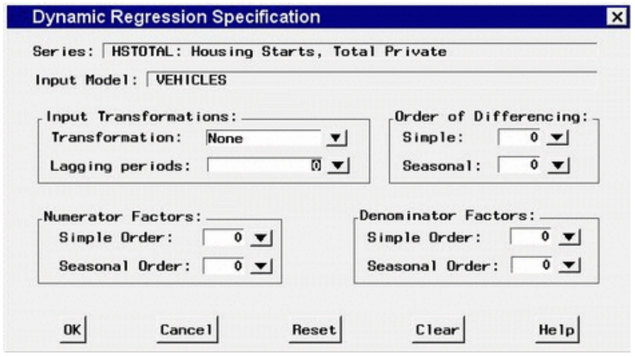
`Remove`
: Deletes a selected lag from the list of differencing lags.

`OK`
: Closes the window and returns the specified list to the Factored ARIMA Model Specification window.

`Cancel`
: Closes the window and discards any lags added to the list.

Dynamic Regression Specification Window

Use the Dynamic Regression Specification window to specify a dynamic regression or transfer function model for the effect of the predictor variable. It is invoked from the Dynamic Regressors Selection window.



Series: HSTOTAL: Housing Starts, Total Private
Input Model: VEHICLES
Input Transformations:
Transformation: None
Lagging periods: 0
Order of Differencing:
Simple: 0
Seasonal: 0
Numerator Factors:
Simple Order: 0
Seasonal Order: 0
Denominator Factors:
Simple Order: 0
Seasonal Order: 0

Controls and Fields

`Series`
: is the name and variable label of the current series.

`Input Model`
: is a descriptive label for the dynamic regression model. You can type a label in this field or allow the system to provide the label. If you leave the label blank, a label is generated automatically based on the options you specify. When no options are specified, the label is the name and variable label of the predictor variable.

`Input Transformation`
: displays the transformation specified for the predictor variable. When a transformation is specified, the transfer function model is fit to the transformed input variable.

`Lagging periods`
: is the pure delay in the effect of the predictor, *l*.

`Simple Order of Differencing`
: is the order of differencing, *d*. Set this field to 1 to use the changes in the predictor variable.

`Seasonal Order of Differencing`
: is the order of seasonal differencing, *D*. Set this field to 1 to difference the predictor variable at the seasonal lags, for example, to use the year-over-year or week-over-week changes in the predictor variable.

`Simple Order Numerator Factors`
: is the order of the numerator factor of the transfer function, *p*.

`Seasonal Order Numerator Factors`
: is the order of the seasonal numerator factor of the transfer function, *P*.

`Simple Order Denominator Factors`
: is the order of the denominator factor of the transfer function, *q*.

`Seasonal Order Denominator Factors`
: is the order of the seasonal denominator factor of the transfer function, *Q*.

`OK`
: closes the window and adds the dynamic regression model specified to the model predictors list.

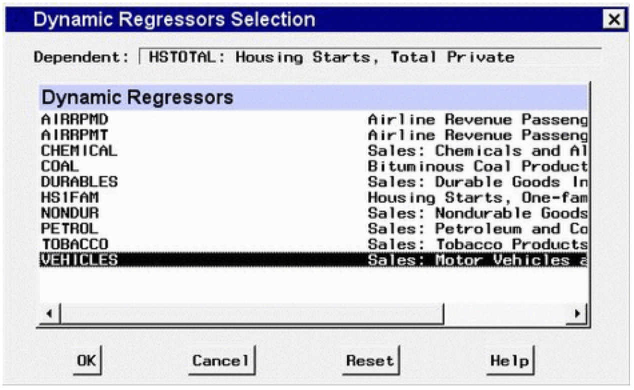
`Cancel`
: closes the window without adding the dynamic regression model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window. This may be useful when editing a predictor specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

Dynamic Regressors Selection Window

Use the Dynamic Regressors Selection window to select an input variable as a dynamic regressor. Access this window from the pop-up menu which appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window.



Dynamic Regressors Selection

Dependent: HSTOTAL: Housing Starts, Total Private

```
Dynamic Regressors
AIRRPMD                                Airline Revenue Passeng
AIRRPMT                                Airline Revenue Passeng
CHEMICAL                               Sales: Chemicals and Al
COAL                                   Bituminous Coal Product
DURABLES                               Sales: Durable Goods In
HS1FAM                                 Housing Starts, One-fam
NONDUR                                 Sales: Nondurable Goods
PETROL                                 Sales: Petroleum and Co
TOBACCO                                Sales: Tobacco Products
VEHICLES                               Sales: Motor Vehicles a
```

Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Dynamic Regressors`
: is a table listing the variables in the input data set. Select one variable in this list as the predictor series.

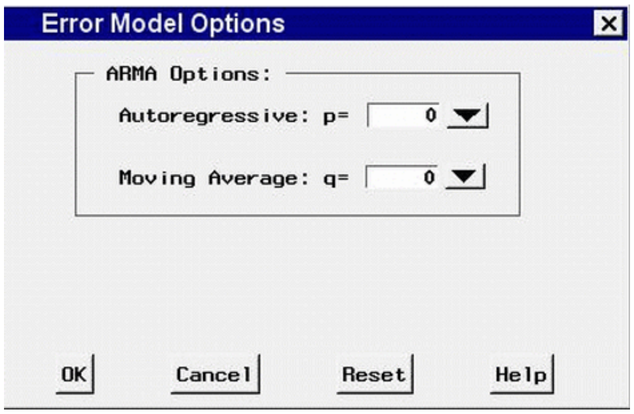
`OK`
: brings up the Dynamic Regression Specification window for you to specify the form of the dynamic regression for the selected predictor series, and then closes the Dynamic Regressors Selection window and adds the specified dynamic regression to the model predictors list.

`Cancel`
: closes the window without adding the dynamic regression model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window.

Error Model Options Window

Use the Error Model Options window to specify the the autoregressive and moving average orders for the residual autocorrelation part of a model defined using the Custom Model Specification window. Access it using the Set button of that window.



Controls and Fields

`ARIMA Options`
: Use these combo boxes to specify the orders of the ARIMA model. You can either type in a value or click the combo box arrow to select from a popup list.

  `Autoregressive`
  : defines the order of the autoregressive part of the model.

  `Moving Average`
  : defines the order of the moving average term.

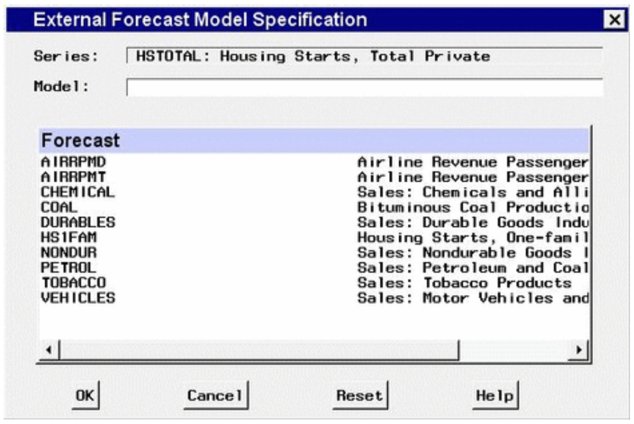
`OK`
: closes the Error Model Options window and returns to the Custom Model Specification window.

`Cancel`
: closes the Error Model Options window and returns to the Custom Model Specification window, discarding any changes made.

`Reset`
: resets all options to their initial values upon entry to the window.

External Forecast Model Specification Window

Use the External Forecast Model Specification window to add to the current project forecasts produced externally to the Time Series Forecasting System. To add an external forecast, select a variable from the selection list and choose the OK button. The name of the selected variable will be added to the list of models fit, and the values of this variable will be used as the forecast. For more information, see "Incorporating Forecasts from Other Sources" in the "Specifying Forecasting Models" chapter.



Series: HSTOTAL: Housing Starts, Total Private

Model:

Forecast	
AIRRPMD	Airline Revenue Passenger
AIRRPMT	Airline Revenue Passenger
CHEMICAL	Sales: Chemicals and Alli
COAL	Bituminous Coal Productio
DURABLES	Sales: Durable Goods Indu
HS1FAM	Housing Starts, One-famil
NONDUR	Sales: Nondurable Goods I
PETROL	Sales: Petroleum and Coal
TOBACCO	Sales: Tobacco Products
VEHICLES	Sales: Motor Vehicles and

Controls and Fields

`OK`
: Closes the window and adds the external forecast to the project.

`Cancel`
: Closes the window without adding an external forecast to the project.

`Reset`
: Deselects any selection made in the selection list.

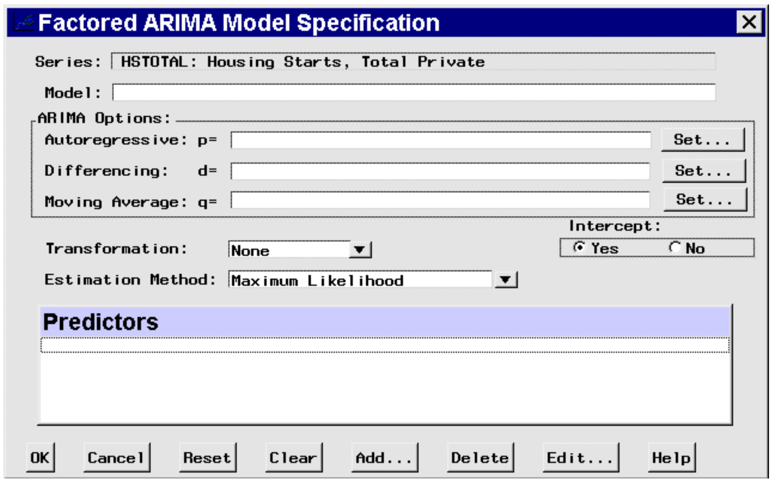
Factored ARIMA Model Specification Window

Use the ARIMA Model Specification window to specify an ARIMA model using the notation:

```
p = (lag, ..., lag) ...  (lag, ..., lag)
d = (lag, ..., lag)
q = (lag, ..., lag) ...  (lag, ..., lag)
```

where p, d, and q represent autoregressive, differencing, and moving average terms, respectively.

Access it from the Develop Models menu, where it is invoked from the Fit Model item under Edit in the menu bar, or from the pop-up menu when you click an empty area of the model table.


Factored ARIMA Model Specification

Series: HSTOTAL: Housing Starts, Total Private

Model:

ARIMA Options:

Autoregressive: p=

Differencing: d=

Moving Average: q=

Intercept: [✓ Yes] No

Transformation: None

Estimation Method: Maximum Likelihood

Predictors

The Factored ARIMA Model Specification window is identical to the ARIMA Model Specification window, except that the p, d, and q terms are specified in a more general and less limited way. Only those controls and fields which differ from the ARIMA Model Specification window are described here.

Controls and Fields

`Model`

is a descriptive label for the model. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the p, d, and q terms that you specify. For example, if you specify `p=(1,2,3), d=(1), q=(12)` and no intercept, the model label is `ARIMA p=(1,2,3) d=(1) q=(12) NOINT`. For monthly data, this is equivalent to the model `ARIMA(3,1,0)(0,0,1)s NOINT` as specified in the ARIMA Model Specification window or the Custom Model Specification window.

`ARIMA Options`

Specifies the ARIMA model in terms of the autoregressive lags (p), differencing lags (d), and moving average lags (q).

`Autoregressive`

defines the autoregressive part of the model. Select the Set button to bring up the AR Polynomial Specification window, where you can add any set of autoregressive lags grouped into any number of factors.

`Differencing`

specifies differencing to be applied to the input data. Select the Set button to bring up the Differencing Specification window, where you can specify any set of differncing lags.

`Moving Average`

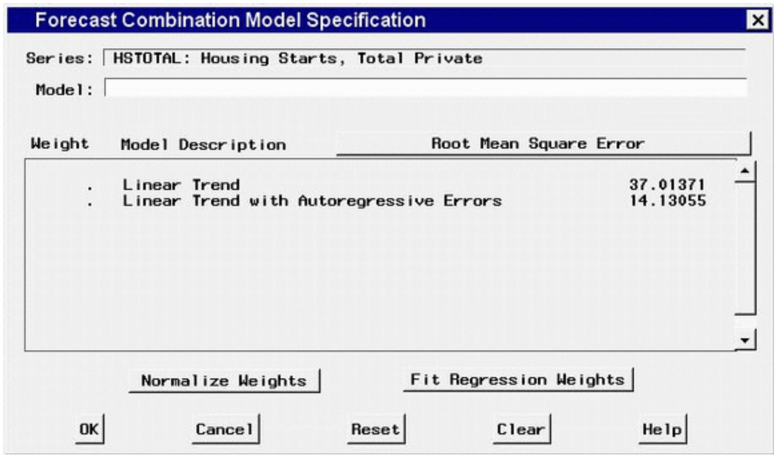
defines the moving average part of the model. Select the Set button to bring up the MA Polynomial Specification window, where you can add any set of moving average lags grouped into any number of factors.

`Estimation Method`

specifies the method used to estimate the model parameters. The Conditional Least Squares and Unconditional Least Squares methods generally require fewer computing resources and are more likely to succeed in fitting complex models. The Maximum Likelihood method requires more resources but provides a better fit in some cases. See also Estimation Details in Chapter 11, “The ARIMA Procedure.”

Forecast Combination Model Specification Window

Use the Forecast Combination Model Specification window to produce forecasts by averaging the forecasts of two or more forecasting models. The specified combination of models is added to the model list for the series. Access this window from the Develop Models window whenever two or more models have been fit to the current series. It is invoked by selecting Combine Forecasts from the Fit Model submenu of the Edit pull-down, or from the pop-up menu which appears when you click an empty part of the model table.



```
Series: HSTOTAL: Housing Starts, Total Private
Model:

Weight   Model Description                            Root Mean Square Error
  .      Linear Trend                                          37.01371
  .      Linear Trend with Autoregressive Errors               14.13055
```

Controls and Fields

`Series`

is the name and variable label of the current series.

`Model`

is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Weight`

is a column of the forecasting model table that contains the weight values for each model. The forecasts for the combined model are computed as a weighted average of the predictions from the models in the table using these weights. Models with missing weight values are not included in the forecast combination. You can type weight values in these fields or you can use other features of the window to set the weights.

`Model Description`

is a column of the forecasting model table that contains the descriptive labels of the forecasting models fit to the current series that are available for combination.

`Root Mean Square Error` (or other statistic name) button
: is the button above the right side of the table. It displays the name of the current model selection criterion: A statistic that measures how well each model in the table fits the values of the current series for observations within the evaluation range. Clicking this button brings up the Model Selection Criterion window to enable you to select a different statistic.

`Normalize Weights` button
: replaces each nonmissing value in the Weights column with the current value divided by the sum of the weights. The resulting weights are proportional to original weights and sum to 1.

`Fit Regression Weights` button
: computes weight values for the models in the table by regressing the series on the predictions from the models. The values in the Weights column are replaced by the estimated coefficients produced by this linear regression. If some weight values are nonmissing and some are missing, only models with nonmissing weight values are included in the regression. If all weights are missing, all models are used.

`OK`
: closes the Forecast Combination Model Specification window and fits the model.

`Cancel`
: closes the Forecast Combination Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the Forecast Combination Model Specification window. This may be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

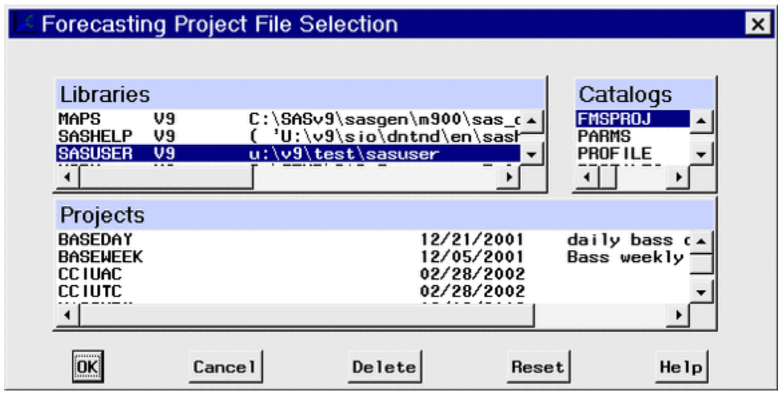
Mouse Button Actions

You can select or deselect models for inclusion in the combination model by positioning the mouse cursor over the model description and pressing the left mouse button. When you select a model in this way, the weights are automatically updated.

The newly selected model is given a weight equal to the average weight of the previously selected models, and all the nonmissing weights are normalized to sum to 1. When you use the mouse to remove a model from the combination, the weight of the deselected model is set to missing and the remaining nonmissing weights are normalized to sum to 1.

Forecasting Project File Selection Window

Use the Forecasting Project File Selection window to locate and load a previously stored forecasting project. Access it from the project Browse button of the Manage Forecasting Project window or the Time Series Forecasting window or from the Open Project item under the File pull-down of the Develop Models window.


Forecasting Project File Selection

Libraries

```
MAPS      V9        C:\SASv9\sasgen\m900\sas_c
SASHELP   V9        (  'U:\v9\sio\dntnd\en\sash
SASUSER   V9        u:\v9\test\sasuser
```

Catalogs

```
FMSPROJ
PARMS
PROFILE
```

Projects

```
BASEDAY                       12/21/2001     daily bass c
BASEWEEK                      12/05/2001     Bass weekly
CCIUAC                        02/28/2002
CCIUTC                        02/28/2002
```

Selection Lists

`Libraries`
: is a list of currently assigned libraries. When you select a library from this list, the catalogs in that library are shown in the catalog selection list.

`Catalogs`
: is a list of catalogs contained in the currently selected library. When you select a catalog from this list, any forecasting project entries stored in that catalog are shown in the projects selection list.

`Projects`
: is a list of forecasting project entries contained in the currently selected catalog.

Controls and Fields

`OK`
: closes the window and opens the selected project.

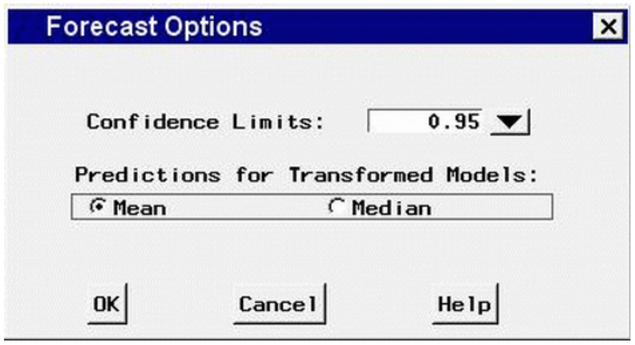
`Cancel`
: closes the window without selecting a project.

`Delete`
: deletes the selected project file.

`Reset`
: restores selections to those which were set before the window was opened.

Forecast Options Window

Use the Forecast Options window to set options to control how forecasts and confidence limits are computed. It is available from the Forecast Options item in the Options pull-down menu of the Develop Models window, Automatic Model Fitting window, Produce Forecasts, and Manage Projects windows.



Controls and Fields

`Confidence Limits`
: specifies the size of the confidence limits for the forecast values. For example, a value of .95 specifies 95% confidence intervals. You can type in a number or select from the popup list.

`Predictions for transformed models`
: These two options control how forecast values are computed for models that employ a series transformation. See the section "Predictions for Transformed Models" in Chapter 41, “Forecasting Process Details,” , for more information. The values are as follows.

  `Mean`
  : specifies that forecast values be predictions of the conditional mean of the series.

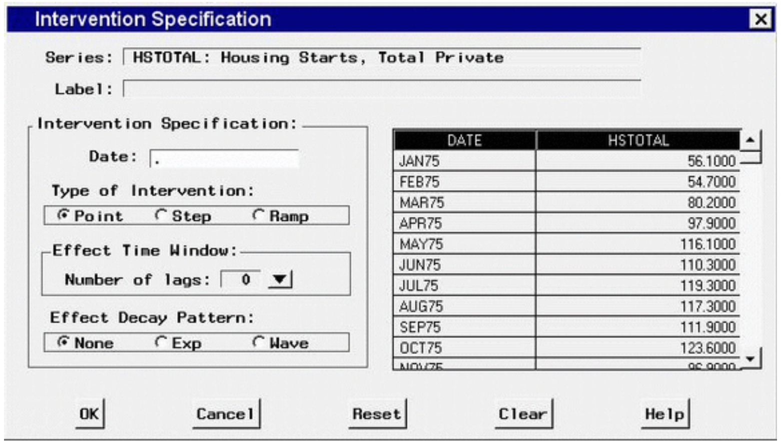
  `Median`
  : specifies that forecast values be predictions of the conditional median of the series.

`OK`
: closes the window and saves the option settings you specified.

`Cancel`
: closes the window without changing the forecast options. Any options you specified are lost.

Intervention Specification Window

Use the Intervention Specification window to specify intervention effects to model the impact on the series of unusual events. Access it from the Intervention for Series window. For more information, see “Interventions” in the “Using Predictor Variables” section.



Series: HSTOTAL: Housing Starts, Total Private

Label:

Intervention Specification:

Date: .

Type of Intervention: [✓ Point] Step Ramp

Effect Time Window:

Number of lags: 0

Effect Decay Pattern: [✓ None] Exp Wave

DATE	HSTOTAL
JAN75	56.1000
FEB75	54.7000
MAR75	80.2000
APR75	97.9000
MAY75	116.1000
JUN75	110.3000
JUL75	119.3000
AUG75	117.3000
SEP75	111.9000
OCT75	123.6000

Controls and Fields

`Series`
: is the name and variable label of the current series.

`Label`
: is a descriptive label for the intervention effect that you specify. You can type a label in this field or allow the system to provide the label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Date`
: is the date that the intervention occurs. You can type a date value in this field, or you can set the date by selecting a row of the data table on the right side of the window.

`Type of Intervention`

- `Point`
  specifies that the intervention variable is zero except for the specified date.
- `Step`
  specifies that the intervention variable is zero before the specified date and a constant 1.0 after the date.
- `Ramp`
  specifies that the intervention variable is an increasing linear function of time after the date of the intervention and zero before the intervention date.

`Number of lags`
: specifies the numerator order for the transfer function model for the intervention effect. Select a value from the popup list.

`Effect Decay Pattern`
: specifies the denominator order for the transfer function model for the intervention effect. The value "Exp" specifies a single lag denominator factor; the value "Wave" specifies a two lag denominator factor.

`OK`
: closes the window and adds the intervention effect specified to the series interventions list.

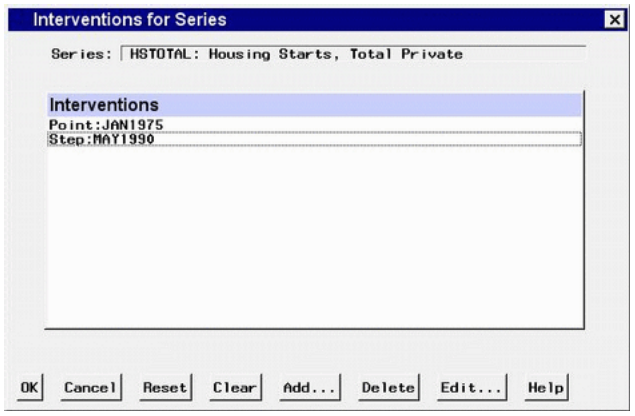
`Cancel`
: closes the window without adding the intervention. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window. This may be useful when editing an intervention specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

Interventions for Series Window

Use the Interventions for Series window to create and edit a list of intervention effects to model the impact on the series of unusual events and to select intervention effects as predictors for forecasting models. Access it from the Add button pop-up menu of the ARIMA Model Specification or Custom Model Specification window, or by selecting Define Interventions from the Tools pull-down in the Develop Models window. For more information, see “Interventions” in Chapter 38, “Using Predictor Variables.”


Interventions for Series

Series: HSTOTAL: Housing Starts, Total Private

Interventions
Point:JAN1975
Step:MAY1990

Controls and Fields

`Series`
: is the name and variable label of the current series.

`OK`
: closes the window. If you access this window from the ARIMA Model Specification window or the Custom Model Specification window, any interventions which are selected (highlighted) in the list are added to the model. If you access this window from the Tools pull-down menu, all interventions in the list are saved for the current series.

`Cancel`
: closes the window without returning a selection or changing the interventions list. Any options you specified are lost.

`Reset`
: resets the list as it was on entry to the window.

`Clear`
: deletes all interventions from the list.

`Add`
: brings up the Intervention Specification window to specify a new intervention effect and add it to the list.

`Delete`
: deletes the currently selected (highlighted) entries from the list.

`Edit`
: brings up the Intervention Specification window to edit the currently selected (highlighted) intervention.

Mouse Button Actions

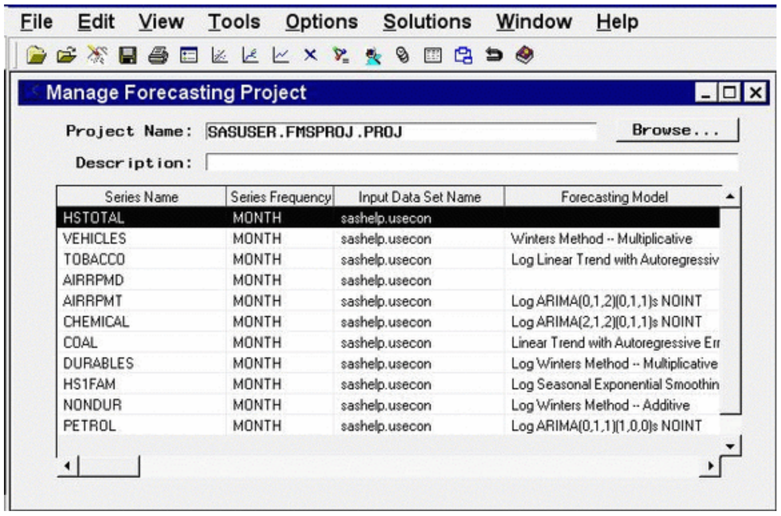
To select or deselect interventions, position the mouse cursor over the intervention's label in the Interventions list and press the left mouse button.

When you position the mouse cursor in the Interventions list and press the right mouse button, a menu containing the actions Add, Delete, and Edit appears. These actions are the same as the Add, Delete, and Edit buttons.

Double-clicking on an intervention in the list invokes an Edit action for that intervention specification.

Manage Forecasting Project Window

Use this resizable window to work with collections of series, models, and options called *projects*. It consists of project name and description fields and a table of information about all the series for which you have fit forecasting models. Access it using the Manage Projects button on the Time Series Forecasting window.



Project Name: SASUSER.FMSPROJ.PROJ
Description:

Series Name	Series Frequency	Input Data Set Name	Forecasting Model
HSTOTAL	MONTH	sashelp.usecon	
VEHICLES	MONTH	sashelp.usecon	Winters Method -- Multiplicative
TOBACCO	MONTH	sashelp.usecon	Log Linear Trend with Autoregressiv
AIRRPMD	MONTH	sashelp.usecon	
AIRRPMT	MONTH	sashelp.usecon	Log ARIMA(0,1,2)(0,1,1)s NOINT
CHEMICAL	MONTH	sashelp.usecon	Log ARIMA(2,1,2)(0,1,1)s NOINT
COAL	MONTH	sashelp.usecon	Linear Trend with Autoregressive Err
DURABLES	MONTH	sashelp.usecon	Log Winters Method -- Multiplicative
HS1FAM	MONTH	sashelp.usecon	Log Seasonal Exponential Smoothin
NONDUR	MONTH	sashelp.usecon	Log Winters Method -- Additive
PETROL	MONTH	sashelp.usecon	Log ARIMA(0,1,1)(1,0,0)s NOINT

Controls and Fields

`Project Name`
: is the name of the SAS catalog entry in which forecasting models and other results will be stored and from which previously stored results are loaded into the forecasting system. You can specify the project by typing a SAS catalog entry name in this field or by selecting the Browse button to the right of this field. If you specify the name of an existing catalog entry, the information in the project file is loaded. If you specify a one level name, it is assumed to be the name of a project in the "fmsproj" catalog in the "sasuser" library. For example, typing `samproj` is equivalent to typing `sasuser.fmsproj.samproj`.

project `Browse` button
: brings up the Forecasting Project File Selection window to enable you to select and load the project from a list of previously stored project files.

`Description`
: is a descriptive label for the forecasting project. The description you type in this field will be stored with the catalog entry shown in the Project field if you save the project.

Series List Table

The table of series for which forecasting models have been fit contains the following columns.

`Series Name`
: is the name of the time series variable represented in the given row of the table.

`Series Frequency`
: is the time interval (data frequency) for the time series.

`Input Data Set Name`
: is the input data set that provided the data for the series.

`Forecasting Model`
: is the descriptive label for the forecasting model selected for the series.

`Statistic Name`
: is the statistic of fit for the forecasting model selected for the series.

`Number of Models`
: is the total number of forecasting models fit to the series. If there is more than one model for a series, use the Model List window to see a list of models.

`Series Label`
: is the variable label for the series.

`Time ID Variable Name`
: is the time ID variable for the input data set for the series.

`Series Data Range`
: is the time range of the nonmissing values of the series.

`Model Fit Range`
: is the period of fit used for the series.

`Model Evaluation Range`
is the evaluation period used for the series.

`Forecast Range`
is the forecast period set for the series.

Menu Bar

`File`

`New`
brings up a dialog which lets you create a new project, assign it a name and description, and make it the active project.

`Open`
brings up a dialog which lets you select and load a previously saved project.

`Close`
closes the Manage Forecasting Project window and returns to the main window.

`Save`
saves the current state of the system (including all the models fit to a series) to the current project catalog entry.

`Save As`
saves the current state of the system with a prompt for the name of the catalog entry in which to store the information.

`Save to Data Set`
saves the current project file information in a SAS data set. The contents of the data set are the same as the information displayed in the series list table.

`Delete`
deletes the current project file.

`Import Data`
is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print`
prints the current project file information.

`Print Setup`
brings up the Print Setup window, which allows you to access your operating system print setup.

`Edit`

`Delete Series`
deletes all models for the selected (highlighted) row of the table and removes the series from the project.

`Clear`
: resets the system, deleting all series and models from the project.

`Reset`
: restores the Manage Forecasting Project window to its initial state.

`View`

`Data Set`
: brings up a Viewtable window to display the input data set for the selected (highlighted) series.

`Series`
: brings up the Time Series Viewer window to display plots of the selected (highlighted) series.

`Model`
: brings up the Model Viewer window to show the current forecasting model for the selected series.

`Forecast`
: brings up the Model Viewer to display plots of the forecasts produced by the forecasting model for the selected (highlighted) series.

`Tools`

`Diagnose Series`
: brings up the Series Diagnostics window to perform the automatic series diagnostic process to determine the kinds of forecasting models appropriate for the selected (highlighted) series.

`List Models`
: brings up the Model List window for the selected (highlighted) series, which displays a list of all the models that you fit for the series. This action is the same as double-clicking the mouse on the table row.

`Generate Data`
: brings up the Time Series Simulation window. This window enables you to simulate ARIMA time series processes and is useful for educational exercises or testing the system.

`Refit Models`

`All Series`
: refits all the models for all the series in the project using data within the current fit range.

`Selected Series`
: refits all the models for the currently highlighted series using data within the current fit range.

`Reevaluate Models`

`All Series`
: reevaluates all the models for all the series in the project using data within the current evaluation fit range.

`Selected Series`
: reevaluates all the models for the currently highlighted series using data within the current evaluation range.

`Options`

`Time Ranges`
: brings up the Time Ranges Specification window to enable you to change the fit and evaluation time ranges and the forecast horizon.

`Default Time Ranges`
: brings up the Default Time Ranges window to enable you to control how the system sets the time ranges for series when you do not explicitly set time ranges with the Time Ranges Specification window. Settings made using this window do not affect series you are already working with–they take effect when you select a new series.

`Model Selection List`
: brings up the Model Selection List editor window. Use this to edit the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

`Statistics of Fit`
: brings up the Statistics of Fit Selection window, which controls which of the available statistics will be displayed.

`Forecast Options`
: brings up the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Column Labels`
: enables you to set long or short column labels. Long labels are used by default.

`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process and displayed by the Model Selection List editor window. When the Include Interventions option is selected, the series interventions are also automatically added to the predictors list when you specify a model in the ARIMA and Custom Models Specification windows.

`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Left Mouse Button Actions

If you select a series in the table by positioning the cursor over the table row and clicking with the left mouse button once, that row of the table is highlighted. Menu bar actions such as Delete Series will apply to the highlighted row of the table.

If you select a series in the table by positioning the cursor over the table row and double-clicking with the left mouse button, the system brings up the Model List window for that series, which displays a list of all the models that you fit for the series. This is the same as the List Models action under Tools in the menu bar.

Right Mouse Button Actions

Clicking the right mouse button invokes a pop-up menu of actions applicable to the highlighted series. The actions in this menu are as follows.

`Delete Series`
: deletes the highlighted series and its models from the project. This is the same as Delete Series in the Edit pull-down menu.

`Refit All Models`
: refits all models attached to the highlighted series using data within the current fit range. This is the same as the Selected Series item under Refit Models in the Tools pull-down menu.

`Reevaluate All Models`
: reevaluates all models attached to the highlighted series using data within the current evaluation range. This is the same as the Selected Series item under Reevaluate Models in the Tools pull-down menu.

`List Models`
: invokes the Model List window. This is the same as List Models under the Tools pull-down menu.

`View Series`
: brings up the Time Series Viewer window to display plots of the highlighted series. This is the same as “Series” in the View pull-down menu.

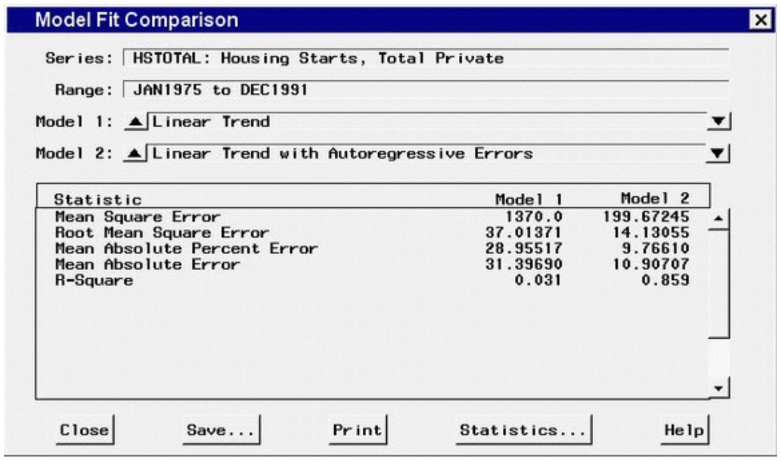
`View Forecasting Model`
: invokes the Model Viewer window to display the forecasting model for the highlighted series. This is the same as “Model” in the View pull-down menu.

`View Forecast`
: brings up the Model Viewer window to display the forecasts for the highlighted series. This is the same as “Forecast” in the View pull-down menu.

`Refresh`
: updates information shown in the Manage Forecasting Project window.

Model Fit Comparison Window

Use the Model Fit Comparison window to compare goodness-of-fit statistics for any two models fit to the current series. Access it from the Tools pull-down menu of the Develop Models window and the Automatic Model Fitting Results window whenever two or more models have been fit to the series.



Series: HSTOTAL: Housing Starts, Total Private

Range: JAN1975 to DEC1991

Model 1: Linear Trend

Model 2: Linear Trend with Autoregressive Errors

```
Statistic                                Model 1      Model 2
Mean Square Error                         1370.0    199.67245
Root Mean Square Error                  37.01371     14.13055
Mean Absolute Percent Error             28.95517      9.76610
Mean Absolute Error                     31.39690     10.90707
R-Square                                   0.031        0.859
```

Controls and Fields

`Series`
: identifies the current time series variable.

`Range`
: displays the starting and ending dates of the series data range.

`Model 1`
: shows the model currently identified as Model 1.

`Model 1 upward arrow button`
: enables you to change the model identified as Model 1 if it is not already the first model in the list of models associated with the series. Select this button to cycle upward through the list of models.

`Model 1 downward arrow button`
: enables you to change the model identified as Model 1 if it is not already the last model in the list of models. Select this button to cycle downward through the list of models.

`Model 2`
: shows the model currently identified as Model 2.

`Model 2 upward arrow button`
: enables you to change the model identified as Model 2 if it is not already the first model in the list of models associated with the series. Select this button to cycle upward through the list of models.

`Model 2 downward arrow button`
: enables you to change the model identified as Model 2 if it is not already the last model in the list of models. Select this button to cycle downward through the list of models.

`Close`
: closes the Model Fit Comparison window.

`Save`
: brings up a dialog for specifying the name and label of a SAS data set to which the statistics will be saved. The data set will contain all available statistics and their values for Model 1 and Model 2, as well as a flag variable that is set to 1 for those statistics that were displayed.

`Print`

prints the contents of the table to the SAS Output window. If you find that the contents do not appear immediately in the Output window, you will need to set scrolling options. Select “Preferences” under the Options submenu of the Tools pull-down. In the Preferences window, select the Advanced tab, then set output scroll lines to a number greater than zero.

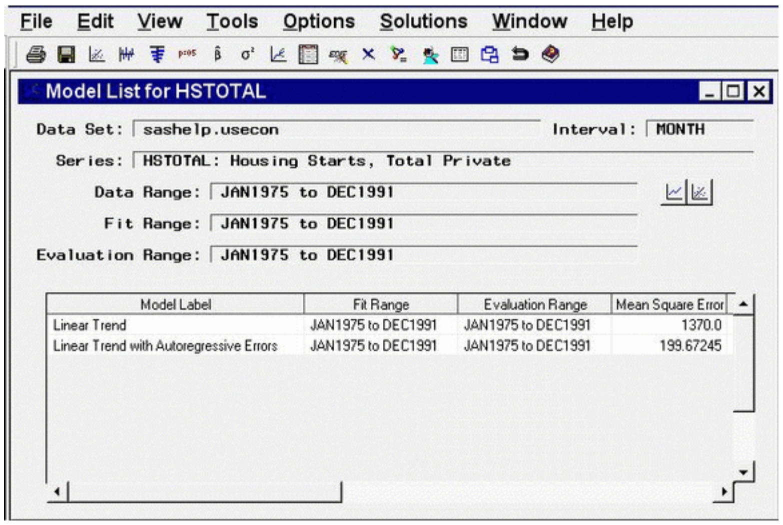
If you want to route the contents to a printer, go to the Output window and select “Print” from the File pull-down menu.

`Statistics`

brings up the Statistics of Fit Selection window for controlling which statistics are displayed.

Model List Window

This resizable window shows all of the models that have been fit to a particular series in a project. Access it from the Manage Forecasting Project window by selecting a series in the series list table and choosing “List Models” from the Tools pull-down menu or by double-clicking the series.



Data Set: sashelp.usecon Interval: MONTH

Series: HSTOTAL: Housing Starts, Total Private

Data Range: JAN1975 to DEC1991

Fit Range: JAN1975 to DEC1991

Evaluation Range: JAN1975 to DEC1991

Model Label	Fit Range	Evaluation Range	Mean Square Error
Linear Trend	JAN1975 to DEC1991	JAN1975 to DEC1991	1370.0
Linear Trend with Autoregressive Errors	JAN1975 to DEC1991	JAN1975 to DEC1991	199.67245

Controls and Fields

`Data Set`

is the name of the current input data set.

`Interval`

is the time interval (data frequency) for the input data set.

`Series`

is the variable name and label of the current time series.

`Data Range`

is the date of the first and last nonmissing data values available for the current series in the input data set.

`Fit Range`

is the current period of fit setting. This is the range of data that will be used to fit models to the series. It may be different from the fit ranges shown in the table, which were in effect when the models were fit.

`Evaluation Range`
: is the current period of evaluation setting. This is the range of data that will be used to calculate the goodness-of-fit statistics for models fit to the series. It may be different from the evaluation ranges shown in the table, which were in effect when the models were fit.

`View Series Graphically` icon
: brings up the Time Series Viewer window to display plots of the current series.

`View Model Graphically` icon
: brings up the Model Viewer to display graphs and tables for the currently highlighted model.

Model List Table

The table of models fit to the series contains columns showing the model label, the fit range and evaluation range used to fit the models, and all of the currently selected fit statistics. You can change the selection of fit statistics using the Statistics of Fit Selection window.

Click on column headings to sort the table by a particular column. If a model is highlighted, clicking with the right mouse button invokes a pop-up menu providing actions applicable to the highlighted model. It includes the following items.

`View Model`
: brings up the Model Viewer on the selected model. This is the same as "Model Predictions" under the View pull-down menu.

`View Parameter Estimates`
: brings up the Model Viewer to display the parameter estimates table for the currently highlighted model. This is the same as "Parameter Estimates" under the View pull-down menu.

`View Statistics of Fit`
: brings up the Model Viewer to display the statistics of fit table for the currently highlighted model. This is the same as "Statistics of FIt" under the View pull-down menu.

`Edit Model`
: brings up the appropriate model specification window for changing the attributes of the highlighted model and fitting the modified model.

`Refit Model`
: refits the highlighted model using the current fit range.

`Reevaluate Model`
: reevaluates the highlighted model using the current evaluation range.

`Delete Model`
: deletes the highlighted model from the project.

`View Forecasts`
: brings up the Model Viewer to show the forecasts for the highlighted model. This is the same as "Forecast Graph" under the View pull-down menu.

Menu Bar

`File`

`Save`
: brings up a dialog which lets you save the contents of the table to a specified SAS data set.

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print`
: sends the contents of the table to a printer as defined through Print Setup.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the window and returns to the Manage Forecasting Projects window.

`Edit`

`Edit Model`
: enables you to modify the specification of the currently highlighted model in the table and fit the modified model. The new model replaces the current model in the table.

`Refit Model`
: refits the currently highlighted model using data within the current fit range.

`Reevaluate Model`
: recomputes statistics of fit for the currently highlighted model using data within the current evaluation range.

`Delete Model`
: deletes the currently highlighted model from the model table.

`Reset`
: restores the contents of the Model List window to the state initially displayed.

`View`

`Series`
: brings up the Time Series Viewer window to display plots of the current series. This is the same as the View Series Graphically icon.

`Model Predictions`
: brings up the Model Viewer to display a predicted and actual plot for the currently highlighted model. This is the same as the View Model Graphically icon.

`Prediction Errors`
: brings up the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
: brings up the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
: brings up the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
: brings up the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
: brings up the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
: brings up the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
: brings up the Model Viewer to display forecasts for the currently highlighted model in a table.

`Options`

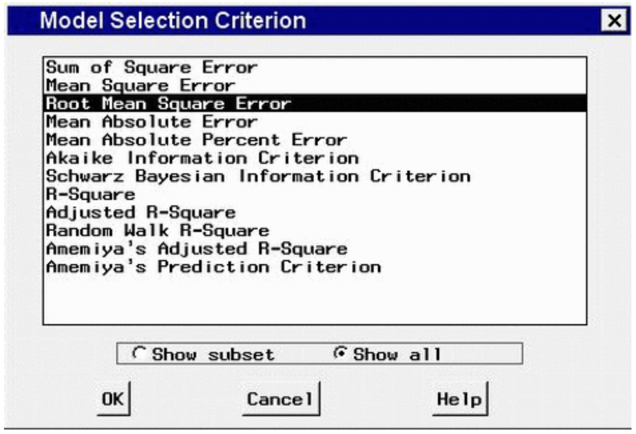
`Statistics of Fit`
: brings up the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Model Viewer, Automatic Model Fitting Results, and Model Fit Comparison windows and available for selection in the Model Selection Criterion menu.

`Column Labels`
: enables you to set long or short column labels. Long labels are used by default.

Model Selection Criterion Window

Use the Model Selection Criterion window to select the model selection criterion statistic used by the automatic selection process to determine the best fitting forecasting model. Model selection criterion statistics are a subset of those shown in the Statistics of Fit Selection window, since some statistics of fit, such as number of observations, are not useful for model selection.

This window is available from the Model Selection Criterion item of the Options pull-down menu of the Develop Models window, Automatic Model Fitting window, and Produce Forecasts window.


Model Selection Criterion
Sum of Square Error
Mean Square Error
Root Mean Square Error
Mean Absolute Error
Mean Absolute Percent Error
Akaike Information Criterion
Schwarz Bayesian Information Criterion
R-Square
Adjusted R-Square
Random Walk R-Square
Amemiya's Adjusted R-Square
Amemiya's Prediction Criterion
Show subset
Show all
OK
Cancel
Help

Show subset [✓ Show all]

Controls and Fields

`Show subset`
: when selected, lists only those model selection criterion statistics which are selected in the Statistics of Fit Selection window.

`Show all`
: when selected, lists all available model selection criterion statistics.

`OK`
: closes the window and sets the model selection criterion to the statistic you specified.

`Cancel`
: closes the window without changing the model selection criterion.

Model Selection List Editor Window

Use the Model Selection List Editor window to edit the model selection list, including adding your own custom models, and to specify which models in the list are to be used in the automatic fitting process. Access it from the Options pull-down menu in the Develop Models, Automatic Model Fitting window, Produce Forecasts, and Manage Projects windows.

The window initially displays the current model list for your project. You can modify this set of models in several ways:

- Open one or more alternate model lists to replace or append to the current model list. These can be either model lists included with the software or model lists previously saved by you or other users.
- Turn the autofit option on or off for individual models. Those which are not flagged for autofit will be available using the Models to Fit window but not using automatic model fitting.
- Delete models from the list which are not needed for your project.
- Reorder the models in the list.

- Edit models in the list.
- Create a new empty list.
- Add new models to the list.

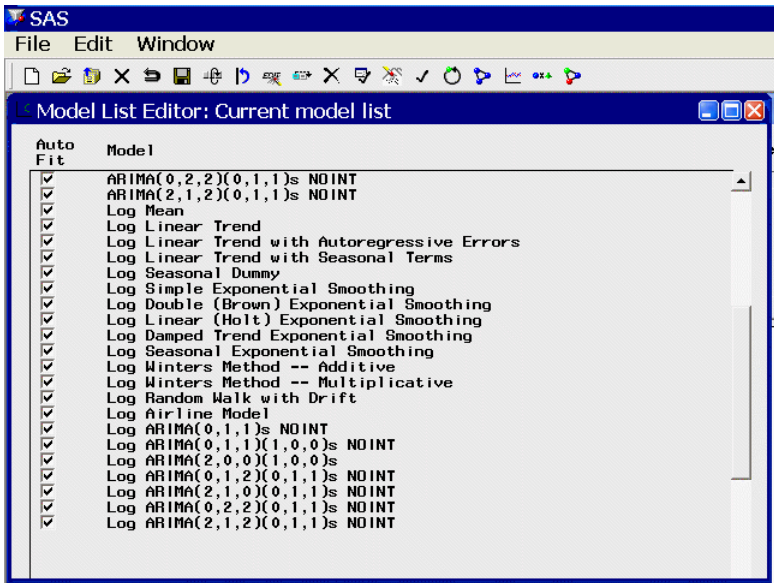
Having modified the current model list, you can save it for future use in several ways:

- Save it in a catalog so it can be opened later in the Model Selection List Editor.
- Save it as the user default to be used automatically when new projects are created.
- Select *close* to close the Model Selection List Editor and attach the modified model selection list to the current project.
- Select *cancel* to close the Model Selection List Editor without changing the current project's model selection list.

Since model selection lists are not bound to specific data sources, care must be taken when including data-specific features such as interventions and regressors. When you add an ARIMA, Factored ARIMA, or Custom model to the list, you can add regressors by selecting from the variables in the current data set. If there is no current data set, you will be prompted to specify a data set so you can select regressors from the series it contains.

If you use a model list having models with a particular regressor name on a data set which does not contain a series of that name, model fitting will fail. However, you can make global changes to the regressor names in the model list using *Set regressor names*. For example, you might use the list of dynamic regression models found in the sashelp.forcast catalog. It uses the regressor name "price." If your regessor series is named "x," you can specify "price" as the current regressor name and "x" as the "change to" name. The change will be applied to all models in the list containing the specified regressor name.

Interventions cannot be defined for models defined from the Model Selection List Editor. However, you can define interventions using the Intervention Specification Window and apply them to your models by turning on the Include Interventions option.



```
Auto
Fit   Model
☑     ARIMA(0,2,2)(0,1,1)s NOINT
☑     ARIMA(2,1,2)(0,1,1)s NOINT
☑     Log Mean
☑     Log Linear Trend
☑     Log Linear Trend with Autoregressive Errors
☑     Log Linear Trend with Seasonal Terms
☑     Log Seasonal Dummy
☑     Log Simple Exponential Smoothing
☑     Log Double (Brown) Exponential Smoothing
☑     Log Linear (Holt) Exponential Smoothing
☑     Log Damped Trend Exponential Smoothing
☑     Log Seasonal Exponential Smoothing
☑     Log Winters Method -- Additive
☑     Log Winters Method -- Multiplicative
☑     Log Random Walk with Drift
☑     Log Airline Model
☑     Log ARIMA(0,1,1)s NOINT
☑     Log ARIMA(0,1,1)(1,0,0)s NOINT
☑     Log ARIMA(2,0,0)(1,0,0)s
☑     Log ARIMA(0,1,2)(0,1,1)s NOINT
☑     Log ARIMA(2,1,0)(0,1,1)s NOINT
☑     Log ARIMA(0,2,2)(0,1,1)s NOINT
☑     Log ARIMA(2,1,2)(0,1,1)s NOINT
```

Auto Fit

The auto fit column of check boxes enables you to eliminate some of the models from being used in the automatic fitting process without having to delete them from the list. By default, all models are checked, meaning that they are all used for automatic fitting.

Model

This column displays the descriptions of all models in the model selection list. You can select one or more models by clicking them with the left mouse button. Selected models are highlighted and become the object of the actions Edit, Move, and Delete.

Menu Bar

`File`

`New`
: creates a new empty model selection list.

`Open`
: brings up a dialog for selecting one or more existing model selection lists to open. If you select multiple lists, they are all opened at once as a concatenated list. This helps you build large specialized model lists quickly by mixing and matching various existing lists such as the various ARIMA model lists included in SASHELP.FORCAST. By default, the lists you open replace the current model list. Select the “append” radio button if you want to append them to the current model list.

`Open System Default`
: opens the default model list supplied with the product.

`Cancel`
: exits the window without applying any changes to the current project's model selection list.

`Close`
: closes the window and applies any changes made to the project's model selection list.

`Save`
: brings up a dialog for saving the edited model selection list in a catalog of your choice.

`Save as User Default`
: saves your edited model list as a default list for new projects. The location of this saved list is shown on the message line. When you create new projects, the system searches for this model list and uses it if it is found. If it is not found, the system uses the original default model list supplied with the product.

`Edit`

`Reset`
: restores the list to its initial state when the window was invoked.

`Add Model`
: enables you to add new models to the selection list. You can use the Smoothing Model Specification window, the ARIMA Model Specification window, the Factored ARIMA Model Specification window, or the Custom Model Specification window.

`Edit Selected`
: brings up the appropriate model specification window for changing the attributes of the highlighted model and adding the modified model to the selection list. The original model is not deleted.

`Move Selected`
: enables you to reorder the models in the list. Select one or more models, then select Move Selected from the menu or toolbar. A note appears on the message line: "Select the row after which the selected models are to be moved." Then select any unhighlighted row in the table. The selected models will be moved after this row.

`Delete`
: deletes any highlighted models from the list. This item is not available if no models are selected.

`Set Regressor Names`
: brings up a dialog for changing all occurrences of a given regressor name in the models of the current model selection list to a name that you specify.

`Select All`
: selects all models in the list.

`Clear Selections`
: deselects all models in the list.

`Select All for Autofit`
: checks the autofit check boxes of all models in the list.

`Clear Autofit Selections`
deselects the autofit check boxes of all models in the list.

Mouse Button Actions

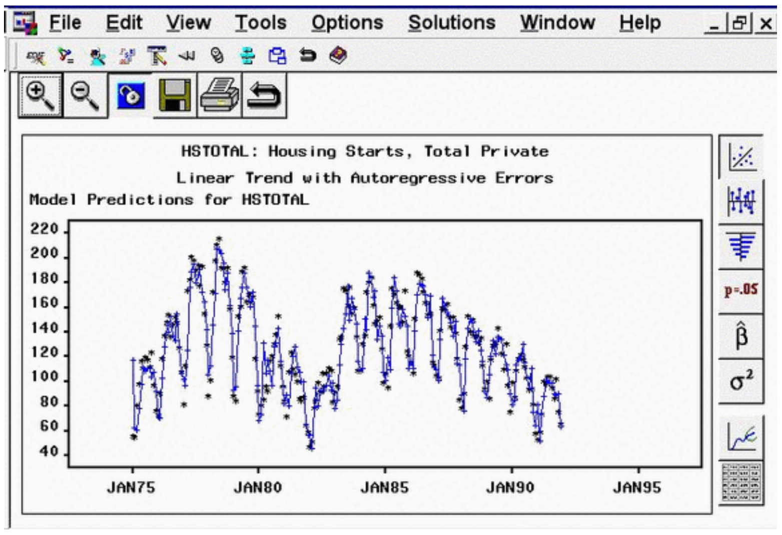
Selecting any model description in the table with the left mouse button selects (highlights) that model. Selecting the same model again deselects it. Multiple selections are allowed.

Selecting the auto fit check box in any row toggles the associated model's eligibility for use in automatic model fitting.

Selecting the right mouse button brings up a pop-up menu.

Model Viewer Window

This resizable window provides plots and tables of actual values, model predictions, forecasts, and related statistics. The various plots and tables available are referred to as *views*. The following "View Selection Icons" section explains how to change the view.


File
Edit
View
Tools
Options
Solutions
Window
Help
HSTOTAL: Housing Starts, Total Private
Linear Trend with Autoregressive Errors
Model Predictions for HSTOTAL
220
200
180
160
140
120
100
80
60
40
JAN75
JAN80
JAN85
JAN90
JAN95
p=.05
β̂
σ²

You can access Model Viewer in a number of ways, including the View Model Graphically icon of the Develop Models and Model List windows, the Graph button of the Automatic Model Fitting Results window, and the Model item under the View pull-down in the Manage Forecasting Project window. In addition, you can go directly to a selected view in the Model Viewer window by selecting Model Predictions, Prediction Errors, Statistics of Fit, Prediction Error Autocorrelations, Prediction Error Tests, Parameter Estimates, Forecast Graph, or Forecast Table from the View pull-down menu or corresponding toolbar icon or popup menu item in the Develop Models, Model List, or Automatic Model Fitting Results windows.

The state of the Model Viewer window is controlled by the current model and the currently selected view. You can resize this window, and you can use other windows without closing the Model Viewer window. By default, the Model

Viewer window is automatically updated to display the new model when you switch to working with another model (that is, when you highlight a different model). You can unlink the Model Viewer window from the current model selection by selecting the Link/Unlink icon from the window's horizontal tool bar. See "Link/Unlink" under "Tool Bar Icons".

For more information, see "Model Viewer" in the "Getting Started" section.

Tool Bar Icons

The Model Viewer window contains a horizontal row of icons called the Tool Bar. Corresponding menu items appear under various pull-down menus. The function of each icon is explained in the following list.

`Zoom in`
: In the Model Predictions, Prediction Errors, and Forecast Graph views, the Zoom In action changes the mouse cursor into cross hairs that you can use with the left mouse button to define a region of the graph to zoom in on. In the Prediction Error Autocorrelations and Prediction Error Tests views, Zoom In reduces the number of lags displayed.

`Zoom out`
: reverses the previous Zoom In action.

`Link/Unlink viewer`
: disconnects or connects the Model Viewer window to the model table (Develop Models window, Model List window, or Automatic Model Fitting Results window). When the viewer is linked, selecting another model in the model table causes the model viewer to be updated to show the selected model. When the Viewer is unlinked, selecting another model does not affect the viewer. This feature is useful for comparing two or more models graphically. You can display a model of interest in the Model Viewer, unlink it, then select another model and bring up another Model Viewer window for that model. Position the viewer windows side by side for convenient comparisons of models, or use the Next Viewer icon or F12 function key to switch between them.

`Save`
: saves the contents of the Model Viewer window. By default, an html page is created. This enables you to display graphs and tables using the Results Viewer or publish them on the Web or your intranet. See also "Save Graph As" and "Save Data As" under "Menu Bar" below.

`Print`
: prints the contents of the viewer window.

`Close`
: closes the Model Viewer window and returns to the window from which it was invoked.

View Selection Icons

At the right hand side of the Model Viewer window is a vertical tool bar to select the view, that is, the kind of plot or table that the viewer displays. Corresponding menu items appear under View in the menu bar. The function of each icon is explained in the following list.

`Model Predictions`
: displays a plot of actual series values and model predictions over time. Click on individual points in the graph to get a display of the type (actual or predicted), ID value, and data value in the upper right corner of the window.

`Prediction Errors`
: displays a plot of model prediction errors (residuals) over time. Click individual points in the graph to get a display of the prediction error value in the upper right corner of the window.

`Prediction Error Autocorrelations`
: displays horizontal bar charts of the sample autocorrelation, partial autocorrelation, and inverse autocorrelation functions for the model prediction errors. Overlaid line plots represent confidence limits computed at plus and minus two standard errors. Click on any of the bars to display its value.

`Prediction Error Tests`
: displays horizontal bar charts representing results of white noise and stationarity tests on the model prediction errors. The first bar chart shows the significance probability of the Ljung-Box chi-square statistic computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the series is white noise. Click on any of the bars to display an interpretation.

  The second bar chart shows tests of stationarity of the model prediction errors, where longer bars favor the conclusion that the series is stationary. Each bar displays the significance probability of the augmented Dickey-Fuller unit root test to the given autoregressive lag. Long bars represent higher levels of significance against the null hypothesis that the series contains a unit root. For seasonal data, a third bar chart appears for seasonal root tests. Click on any of the bars to display an interpretation.

`Parameter Estimates`
: displays a table showing model parameter estimates along with standard errors and *t*-tests for the null hypothesis that the parameter is zero.

`Statistics of Fit`
: displays a table of statistics of fit for the selected model. The set of statistics shown can be changed using the Statistics of Fit item under Options in the menu bar.

`Forecast Graph`
: displays a plot of actual and predicted values for the series data range, followed by a horizontal reference line and forecasted values with confidence limits. Click on individual points in the graph to get a display of the type, date/time, and value of the data point in the upper right corner of the window.

`Forecast Table`
: displays a data table with columns containing the date/time, actual, predicted, error (residual), lower confidence limit, and upper confidence limit values, together with any predictor series.

Menu Bar

`File`

`Save Graph`
: saves the plot displayed in viewer window as a SAS/GRAPH grseg catalog entry. When the current view is a table, this menu item is not available. See also "Save" under "Tool Bar Icons". If a graphics catalog entry name has not already been specified, this action functions like "Save Graph As".

`Save Graph As`
: saves the current graph as a SAS/GRAPH grseg catalog entry in a SAS catalog that you specify and/or as an Output Delivery System (ODS) object. By default, an html page is created, with the graph embedded as a gif image.

`Save Data`
: saves the data displayed in the viewer window in a SAS data set, where applicable.

`Save Data As`
: saves the data in a SAS data set that you specify and/or as an Output Delivery System (ODS) object. By default, an html page is created, with the data displayed as a table.

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Graph`
: prints the contents of the viewer window if the current view is a graph. This is the same as the Print tool bar icon. If the current view is a table, this menu item is not available.

`Print Data`
: prints the data displayed in the viewer window, where applicable.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Print Preview`
: brings up a preview window to show how your plots will appear when printed.

`Close`
: closes the Model Viewer window and returns to the window from which it was invoked.

`Edit`

`Edit Model`
: enables you to modify the specification of the current model and to fit the modified model, which is then displayed in the viewer.

`Refit Model`
: refits the current model using data within the current fit range. This action also causes the ranges to be reset if the data range has changed.

`Reevaluate Model`
: reevaluates the current model using data within the current evaluation range. This action also causes the ranges to be reset if the data range has changed.

`View`

See the "View Selection Icons" section. It describes each of the items available under "View", except "Zoom Way Out".

`Zoom Way Out`
: zooms the plot out as far as it will go, undoing all prior zoom in operations.

`Tools`

`Link Viewer`
: See "Link/Unlink" under "Tool Bar Icons".

`Options`

`Time Ranges`
: brings up the Time Ranges Specification window to enable you to change the period of fit, period of evaluation, or forecast horizon to be applied to subsequently fit models.

`Statistics of Fit`
: brings up the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the statistics of fit table and available for selection in the Model Selection Criterion menu.

`Forecast Options`
: brings up the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Residual Plot Options`
: Provides a choice of four methods of computing prediction errors for models which include a data transformation.

  `Prediction Errors`
  : computes the difference between the transformed series actual values and model predictions.

  `Normalized Prediction Errors`
  : computes prediction errors in normalized form.

  `Model Residuals`
  : computes the difference between the untransformed series values and the untransformed model predictions.

`Normalized Model Residuals`
computes model residuals in normalized form.

`Number of Lags`
brings up a window to enable you to specify the number of lags shown in the Prediction Error Autocorrelations and Prediction Error Tests views. You can also use the Zoom In and Zoom Out actions to control the number of lags displayed.

`Correlation Probabilities`
controls whether the bar charts in the Prediction Error Autocorrelations view represent significance probabilities or values of the correlation coefficient. A check mark or filled check box next to this item indicates that significance probabilities are displayed. In each case the bar graph horizontal axis label changes accordingly.

`Include Interventions`
controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Mouse Button Actions

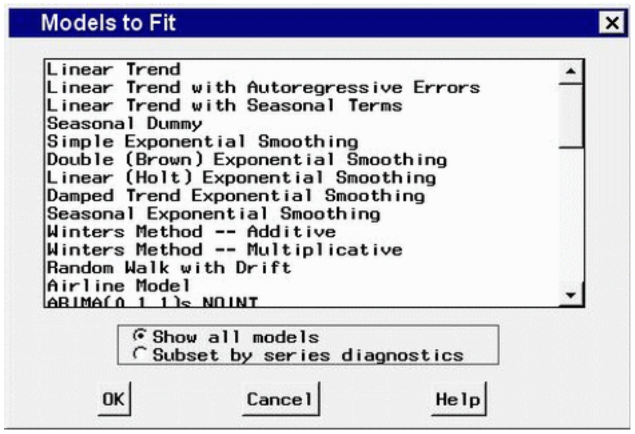
You can examine the data values of individual points in the Model Predictions, Model Prediction Errors, and Forecast Graph views of the Model Viewer by positioning the mouse cursor over the point and clicking the left mouse button. The date/time and data values as well as the type (actual, predicted, and so forth) are displayed in a box that appears in the upper right corner of the Viewer window. Click the mouse elsewhere or select any action to dismiss the data box.

Similarly, you can display values in the Prediction Error Autocorrelations view by clicking on any of the bars. Clicking on bars in the Prediction Error Tests view displays a Recommendation for Current View window which explains the test represented by the bar.

When you select the Zoom In action in the Predicted Values, Model Prediction Errors, and Forecasted Values views, you can use the mouse to define a region of the graph to zoom. Position the mouse cursor at one corner of the region, press the left mouse button, and move the mouse cursor to the opposite corner of the region while holding the left mouse button down. When you release the mouse button, the plot is redrawn to show an expanded view of the data within the region you selected.

Models to Fit Window

Use the Models to Fit window to fit models by choosing them from the current model selection list. Access it using “Fit Models from List” under the Fit Model submenu of the Edit pull-down in the Develop Models window, or the pop-up menu which appears when you click an empty area of the model table in the Develop Models window. If you want to alter the list of models which appears here, use the Model Selection List editor window.


Models to Fit
Linear Trend
Linear Trend with Autoregressive Errors
Linear Trend with Seasonal Terms
Seasonal Dummy
Simple Exponential Smoothing
Double (Brown) Exponential Smoothing
Linear (Holt) Exponential Smoothing
Damped Trend Exponential Smoothing
Seasonal Exponential Smoothing
Winters Method -- Additive
Winters Method -- Multiplicative
Random Walk with Drift
Airline Model
ARIMA(0,1,1)s NOINT
Show all models
Subset by series diagnostics
OK
Cancel
Help

To select a model to be fit, use the left mouse button. To select more than one model to fit, drag with the mouse, or select the first model, then press the shift key while selecting the last model. For noncontiguous selections, press the control key while selecting with the mouse. To begin fitting the models, double-click the last selection or select the OK button.

If series diagnostics have been performed, the radio box is ungrayed. If the Subset by series diagnostics radio button is selected, only those models in the selection list that fit the diagnostic criteria will be shown for selection. If you wish to choose models that do not fit the diagnostic criteria, select the Show all models button.

Controls and Fields

`Show all models`

when selected, lists all available models, regardless of the setting of the series diagnostics options.

`Subset by series diagnostics`

when selected, lists only the available models that are consistent with the series diagnostics options.

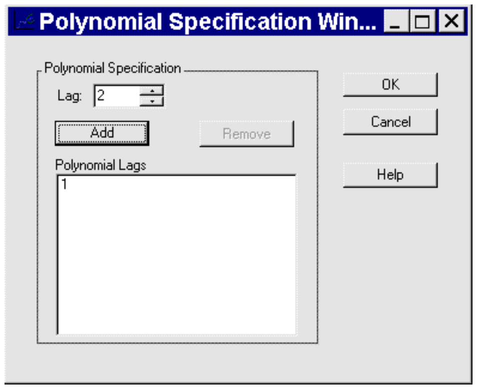
`OK`

closes the Models to Fit window and fits the selected models.

`Cancel`
: closes the window without fitting any models. Any selections you made are lost.

Polynomial Specification Window

Use the Polynomial Specification window to add a polynomial to an ARIMA model. The set of lags defined here become a polynomial factor, denoted by a list of lags in parentheses, when you select “OK”. If you accessed this window from the AR Polynomial Specification window it is added to the autoregressive part of the model. If you accessed it from the MA Polynomial Specification window it is added to the moving average part of the model.



Controls and Fields

`Lag`
: specifies a lag value to add to the list. Type in a positive integer or select one by clicking the spin box arrows.

`Add`
: adds the value in the Lag spin box to the list of polynomial lags. Duplicate values are not allowed.

`Remove`
: deletes a selected lag from the list of polynomial lags.

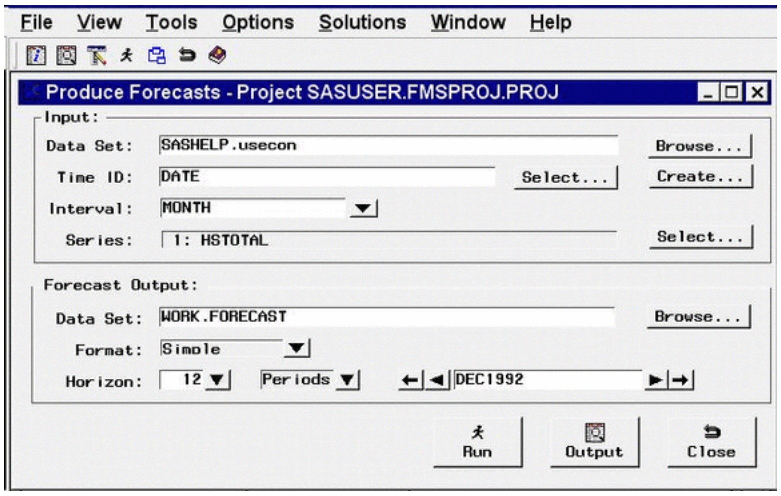
`Polynomial Lags`
: is a list of unique integers representing lags to be added to the model.

`OK`
: closes the window and returns the specified polynomial to the AR or MA polynomial specification window.

`Cancel`
: closes the window and discards any polynomial lags added to the list.

Produce Forecasts Window

Use the Produce Forecasts window to produce forecasts for the series in the current input data set for which you have fit forecasting models. Access it using the Produce Forecasts button of the Time Series Forecasting window.



Produce Forecasts - Project SASUSER.FMSPROJ.PROJ

Input:

Data Set: SASHELP.usecon

Time ID: DATE

Interval: MONTH

Series: 1: HSTOTAL

Forecast Output:

Data Set: WORK.FORECAST

Format: Simple

Horizon: 12 Periods DEC1992

Controls and Fields

Input `Data Set`

is the name of the current input data set. To specify the input data set, you can type a one or two level SAS data set name in this field or select the Browse button to the right of the field.

Input data set `Browse` button

brings up the Data Set Selection window to enable you to select the input data set.

`Time ID`

is the name of the time ID variable for the input data set. To specify this variable, you can type the ID variable name in this field or use the Select button.

Time ID `Select` button

brings up the Time ID Variable Specification window.

`Create` button

brings up a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`

is the time interval between observations (data frequency) in the current input data set. If the interval is not automatically filled in by the system, you can type in an interval name here, or select one from the popup list.

`Series`

indicates the number and names of time series variables for which forecasts will be produced.

Series `Select` button

brings up the Series to Process window to let you select the series for which you want to produce forecasts.

Forecast Output `Data Set`

is the name of the output data set that will contain the forecasts. Type the name of the output data set in this field or click the Browse button.

Forecast Output `Browse` button
: brings up a dialog to let you locate an existing data set to which to save the forecasts.

`Format`
: Lets you select one of three formats for the forecast data set:

  `Simple`
  : specifies the simple format for the output data set. The data set contains the time ID variable and the forecast variables and contains one observation per time period. Observations for earlier time periods contain actual values copied from the input data set; later observations contain the forecasts.

  `Interleaved`
  : specifies the interleaved format for the output data set. The data set contains the time ID variable, the variable TYPE, and the forecast variables. There are several observations per time period, with the meaning of each observation identified by the TYPE variable.

  `Concatenated`
  : specifies the concatenated format for the output data set. The data set contains the variable SERIES, the time ID variable, and the variables ACTUAL, PREDICT, ERROR, LOWER, and UPPER. There is one observation per time period per forecast series. The variable SERIES contains the name of the forecast series, and the data set is sorted by SERIES and DATE.

`Horizon`
: is the number of periods or years to forecast beyond the end of the input data range. To specify the forecast horizon, you can type a value in this field or select one from the popup list.

`Horizon periods`
: selects the units to apply to the horizon. By default, the horizon value represents number of periods. For example, if the interval is month, the horizon represents the number of months to forecast. Depending on the interval, you can also select weeks or years, so that the horizon is measured in those units.

`Horizon date`
: is the ending date of the forecast horizon. You can type in a date using a form recognized by a SAS date informat, or you can increment or decrement the date shown using the left and right arrows. The outer arrows change the date by a larger amount than the inner arrows. The date field and the horizon field reset each other, so you can use either one to specify the forecasting horizon.

`Run` button
: produces forecasts for the selected series and stores the forecasts in the specified output SAS data set.

`Output` button
: brings up a Viewtable window to display the output data set. This button becomes ungrayed once the forecasts have been written to the data set.

`Close` button
: closes the Produce Forecasts window and returns to the Time Series Forecasting window.

Menu Bar

`File`

`Import Data`
: is available if you license SAS/Access software. It brings up an Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the Produce Forecasts window and returns to the Time Series Forecasting window.

`View`

`Input Data Set`
: brings up a Viewtable window to browse the current input data set.

`Output Data Set`
: brings up a Viewtable window to browse the output data set. This is the same as the Output button.

`Tools`

`Produce Forecasts`
: produces forecasts for the selected series and stores the forecasts in the specified output SAS data set. This is the same as the Run button.

`Options`

`Default Time Ranges`
: brings up the Default Time Ranges window to enable you to control how the system sets the time ranges when new series are selected.

`Model Selection List`
: brings up the Model Selection List editor window. Use this to edit the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

`Model Selection Criterion`
: brings up the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model.

`Statistics of Fit`
: brings up the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Statistics of Fit table and available for selection in the Model Selection Criterion window.

`Forecast Options`
: brings up the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Forecast Data Set`
: lets you select one of three formats for the forecast data set. See the description under "format".

`Alignment of Dates`

`Beginning`
: aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

`Middle`
: aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

`End`
: aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
: brings up the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Set Toolbar Type`

`Image Only`
: displays the tool bar items as icons without text.

`Label Only`
: displays the tool bar items as text without icon images.

`Both`
: displays the tool bar items as both text and icon images.

`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

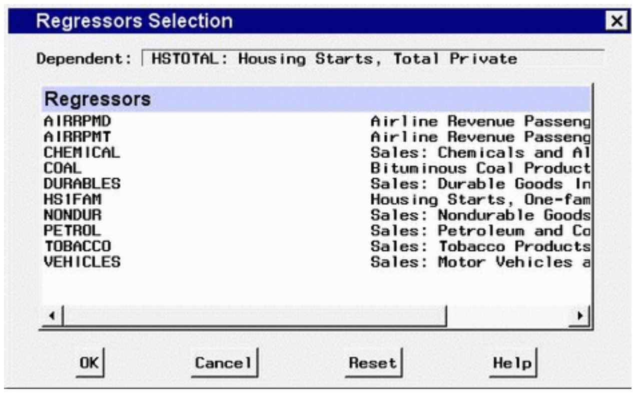
`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: controls whether SAS statements submitted by the forecasting system are

printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Regressors Selection Window

Use the Regressors Selection window to select one or more time series variables in the input data set to include as regressors in the forecasting model to predict the dependent series. Access it from the pop-up menu which appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window.



Dependent: HSTOTAL: Housing Starts, Total Private

Regressors	
AIRRPMD	Airline Revenue Passeng
AIRRPMT	Airline Revenue Passeng
CHEMICAL	Sales: Chemicals and Al
COAL	Bituminous Coal Product
DURABLES	Sales: Durable Goods In
HS1FAM	Housing Starts, One-fam
NONDUR	Sales: Nondurable Goods
PETROL	Sales: Petroleum and Co
TOBACCO	Sales: Tobacco Products
VEHICLES	Sales: Motor Vehicles a

Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Regressors`
: is a table listing the names and labels of the variables in the input data set available for selection as regressors. The variables that you select are highlighted. Selecting a highlighted row again deselects that variable.

`OK`
: closes the Regressors Selection window and adds the selected variables as regressors in the model.

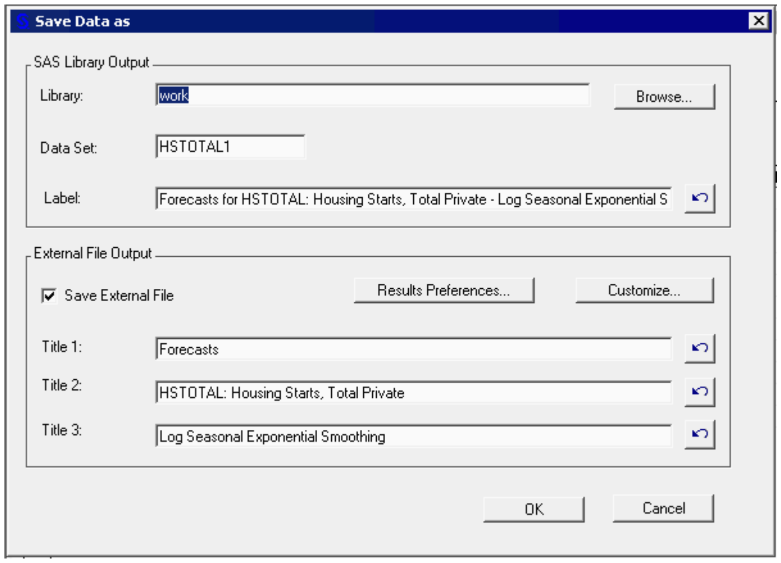
`Cancel`
: closes the window without adding any regressors. Any selections you made are lost.

`Reset`
: resets all options to their initial values upon entry to the window.

Save Data As

Use Save Data As from the Time Series Viewer Window or the Model Viewer Window to save data displayed in a table to a SAS data set or external file.

Use Save Forecast As from the Develop Models Window to save forecasts and related data including the series name, model, and interval. It supports append mode, enabling you to accumulate the forecasts of multiple series in a single data set.


Save Data as

SAS Library Output

Library: work

Data Set: HSTOTAL1

Label: Forecasts for HSTOTAL: Housing Starts, Total Private - Log Seasonal Exponential S

External File Output

[✓] Save External File

Title 1: Forecasts

Title 2: HSTOTAL: Housing Starts, Total Private

Title 3: Log Seasonal Exponential Smoothing

To save your data in a SAS data set, provide a library name or assign one using the Browse button, then provide a data set name or accept the default. Enter a descriptive label for the data set in the Label field. Click OK to save the data set. If you specify an existing data set, it will be overwritten, except in the case of Save Forecast As, described above.

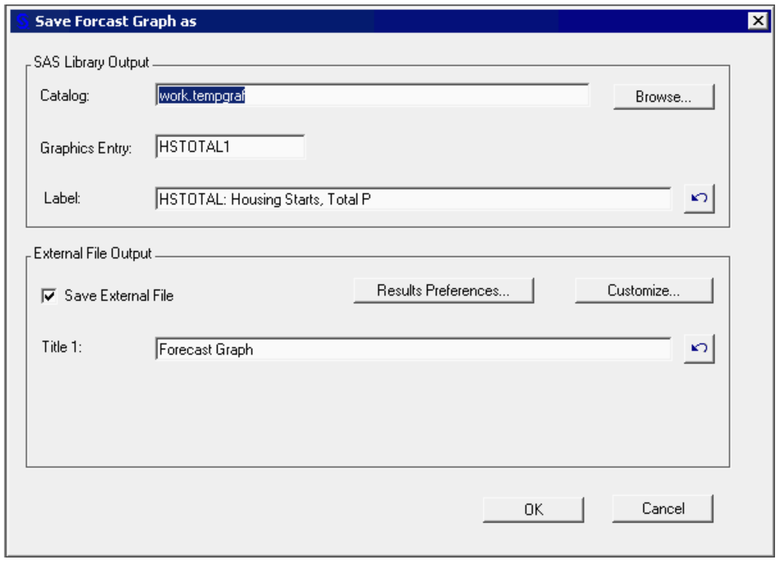
External file output takes advantage of the Output Delivery System (ODS) and is designed primarily for creating html tables for Web reporting. You can build a set of Web pages quickly and use the ODS Results window to view and organize them. To use this feature, check Save External File in the External File Output box. To set ODS options, click Results Preferences, then select the Results tab in the Preferences dialog.

If you have previously saved data of the current type, the system remembers your previous labels and titles. To reuse them, click the arrow button to the right of each of these window fields.

Use the Customize button if you need to specify the name of a custom macro containing ODS statements. You will notice that the default macro simply runs the PRINT procedure. A custom macro can be used to add PRINT procedure and/or ODS statements to customize the type and organization of output files produced.

Save Graph As

Use Save Graph As from the Time Series Viewer Window or the Model Viewer Window to save any of the graphs in a catalog or external file.



Save Forcast Graph as

SAS Library Output

Catalog: work.tempgraf

Graphics Entry: HSTOTAL1

Label: HSTOTAL: Housing Starts, Total P

External File Output

[✓] Save External File

Title 1: Forecast Graph

To save your graph as a grseg catalog entry, enter a two level name for the catalog or select Browse to bring up an Open dialog. Use it to select an existing library or assign a new one and then select a catalog to contain the graph. Click the Open button to open the catalog and close the dialog. Then enter a graphics entry name (eight characters or less) and a label or accept the defaults and click the OK button to save the graph.

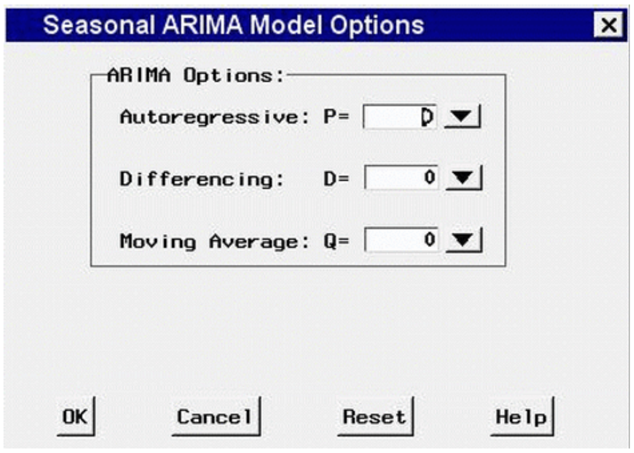
External file output takes advantage of the Output Delivery System (ODS) and is designed primarily for creating gif images and html for Web reporting. You can build a set of Web pages containing graphs and use the Results window to view and organize them. To use this feature, check Save External File in the External File Output box. To set ODS options, click Results Preferences, then select the Results tab in the Preferences dialog.

If you have previously saved graphs of the current type, the system remembers your previous labels and titles. To reuse them, click the arrow button to the right of each of these window fields.

Use the Customize button if you need to specify the name of a custom macro containing ODS statements. You will notice that the default macro simply runs the GREPLAY procedure. Users familiar with ODS may wish to add statements to the macro to customize the type and organization of output files produced.

Seasonal ARIMA Model Options Window

Use the Seasonal ARIMA Model Options window to specify the the autoregressive, differencing, and moving average orders for the seasonal part of a model defined using the Custom Model Specification window. Access it by selecting "Seasonal ARIMA..." from the Seasonal Model combo box of that window.



Controls and Fields

`ARIMA Options`

Use these combo boxes to specify the orders of the ARIMA model. You can either type in a value or click the combo box arrow to select from a popup list.

`Autoregressive`

defines the order of the seasonal autoregressive part of the model.

`Differencing`

defines the order of seasonal differencing.

`Moving Average`

defines the order of the seasonal moving average term.

`OK`

closes the Seasonal ARIMA Model Options window and returns to the Custom Model Specification window.

`Cancel`

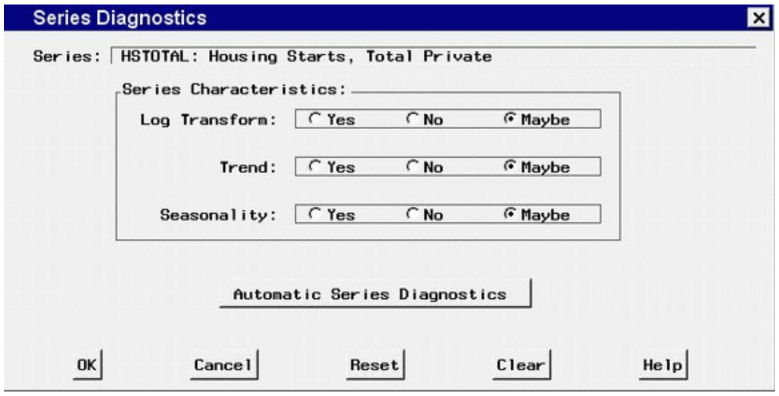
closes the Seasonal ARIMA Model Options window and returns to the Custom Model Specification window, discarding any changes made.

`Reset`

resets all options to their initial values upon entry to the window.

Series Diagnostics Window

Use the Series Diagnostics window to set options to limit the kinds of forecasting models considered for the series according to series characteristics. Access it by selecting "Diagnose Series" from the Tools pull-down menu in the Develop Models, Manage Project, and Time Series Viewer window menu bars. You can let the system diagnose the series' characteristics automatically or you can specify series characteristics according to your judgment using the radio buttons.



Series Diagnostics

Series: HSTOTAL: Housing Starts, Total Private

Series Characteristics:

Log Transform: Yes No [✓ Maybe]

Trend: Yes No [✓ Maybe]

Seasonality: Yes No [✓ Maybe]

For each of the options Log Transform, Trend, and Seasonality, the value "Yes" means that only models appropriate for series with that characteristic should be considered. The value "No" means that only models appropriate for series without that characteristic should be considered. The value "Maybe" means that models should be considered without regard for that characteristic.

Controls and Fields

`Series`
: is the name and variable label of the current series.

`Series Characteristics`

  `Log Transform`
  : specifies whether forecasting models with or without a logarithmic series transformation are appropriate for the series.

  `Trend`
  : specifies whether forecasting models with or without a trend component are appropriate for the series.

  `Seasonality`
  : specifies whether forecasting models with or without a seasonal component are appropriate for the series.

`Automatic Series Diagnostics`
: performs the automatic series diagnostic process. The options Log Transform, Trend, and Seasonality are set according to the results of statistical tests.

`OK`
: closes the Series Diagnostics window.

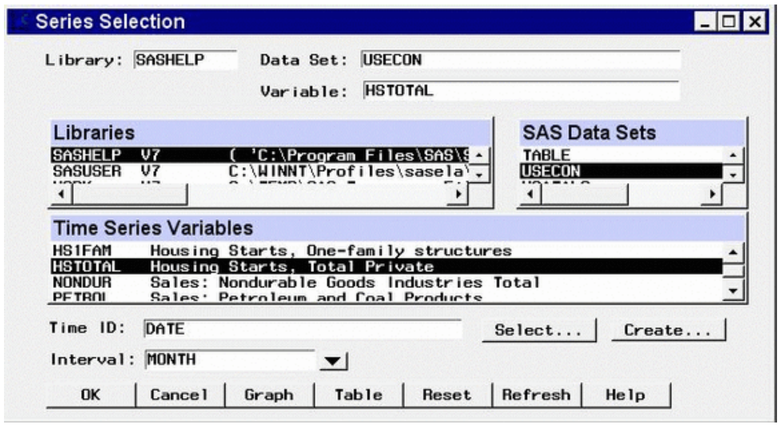
`Cancel`
: closes the Series Diagnostics window without changing the series diagnostics options. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the Series Diagnostics window.

`Clear`
: resets all options to their default values.

Series Selection Window

Use this resizable window to select a time series variable by specifying a library, a SAS data set or view, and a variable. These selections can be made by typing, by selecting from lists, or by a combination of the two. In addition, you can control the time ID variable and time interval, and you can browse the data set or view plots of the series from this window.



Library: SASHELP
Data Set: USECON
Variable: HSTOTAL

```
SASHELP  V7        ( 'C:\Program Files\SAS\S
SASUSER  V7        C:\WINNT\Profiles\sasela\
```

```
TABLE
USECON
```

```
HS1FAM    Housing Starts, One-family structures
HSTOTAL   Housing Starts, Total Private
NONDUR    Sales: Nondurable Goods Industries Total
PETROL    Sales: Petroleum and Coal Products
```

Time ID: DATE
Interval: MONTH

This window appears automatically when you select the View Series Graphically or Develop Models buttons in the Time Series Forecasting window and no series has been selected, and when you bring up the Time Series Viewer as a stand-alone tool. It is also invoked using the Browse button in the Develop Models window.

The system requires that series names be unique for each frequency (interval) within the forecasting project. If you select a series from the current input data set that already exists in the project with the same interval but a different input data set name, the system warns you and gives you the option to cancel the selection, to refit all models associated with the series using the data from the current input data set, to delete the models for the series, or to inherit the existing models.

Controls and Fields

`Library`
: is a SAS libname assigned within the current SAS session. If you know the libname associated with the data set of interest, you can type it in this field and press Return. If it is a valid choice, it will appear in the libraries list and will be highlighted. The SAS Data Sets list will be populated with data sets associated with that libname.

`Data Set`
: is the name of a SAS data set (data file or data view) that resides under the selected libname. If you know the name, you can type it in and press Return. If it is a valid choice, it will appear in the SAS Data Sets list and

will be highlighted, and the Time Series Variables list will be populated with the numeric variables in the data set.

`Variable`
: is the name of a numeric variable contained in the selected data set. You can type the variable name in this field or you can select the variable with the mouse from the Time Series Variables list.

`Time ID`
: is the name of the ID variable for the input data set. To specify the ID variable, you can type the ID variable name in this field or click the Select button.

`Select` button
: brings up the Time ID Variable Specification window to let you select an existing variable in the data set as the Time ID.

`Create` button
: brings up a menu of methods for creating a time ID variable for the input data set. Use this feature if the data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the selected data set. If the interval is not automatically filled in by the system, you can type in an interval name or select one from the popup list. For more information on intervals, see Chapter 3, "Date Intervals, Formats, and Functions," in this book.

`OK`
: This button is present when you have selected "Develop Models" from the Time Series Forecasting window. It closes the Series Selection window and makes the selected series the current series.

`Close`
: If you have selected the View Series Graphically icon from the Time Series Forecasting window, this button returns you to that window. If you have selected a series, it remains selected as the current series.

  If you are using the Time Series Viewer as a stand-alone application, this button closes the application.

`Cancel`
: This button is present when you have selected "Develop Models" from the Time Series Forecasting window. It closes the Series Selection window without applying any selections made.

`Reset`
: resets the fields to their initial values at entry to the window.

`Table`
: brings up a Viewtable window for browsing the selected data set. This can assist you in locating the variable containing data you are looking for.

`Graph`
: brings up the Time Series Viewer window to display the selected time series variable. You can switch to a different series in the Series Selection window without closing the Time Series Viewer window. Position the windows so

they are both visible, or use the Next Viewer tool bar icon or F12 function key to switch between windows.

`Refresh`
: updates all fields and lists on the window. If you assign a new libname without exiting the Series Selection window, use the refresh action to update the Libraries list so that it will include the newly assigned libname. Also use the Refresh action to update the variables list if the input data set is changed.

Selection Lists

`Libraries`
: displays a list of currently assigned libnames. You can select a libname by clicking it with the left mouse button, which is equivalent to typing its name in the Library field. If you cannot locate the library or directory you are interested in, go to the SAS Explorer window, select “New” from the File pull-down menu, then select “Library” and “OK”. This brings up the New Library dialog window. You also assign a libname by submitting a libname statement from the Editor window. Select the Refresh button to make the new libname available in the libraries list.

`SAS Data Sets`
: displays a list of the SAS data sets (data files or data views) located under the selected libname. You can select one of these by clicking with the left mouse button, which is equivalent to typing its name in the Data Set field.

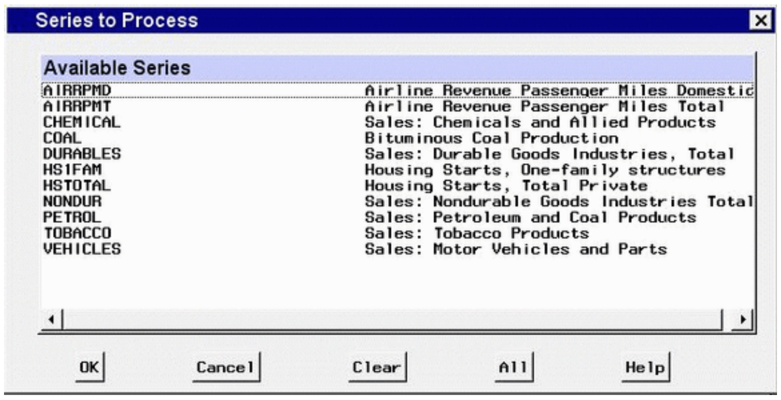
`Time Series Variables`
: displays a list of numeric variables contained within the selected data set. You can select one of these by clicking with the left mouse button, which is equivalent to typing its name in the Variable field. You can double-click on a series to select it and exit the window.

Series to Process Window

Use the Series to Process window to select series for model fitting or forecasting. Access it using the Select button in the Automatic Model Fitting and Produce Forecasts windows. Hold down the shift key or drag with the left mouse button for multiple selections. Use the control key for noncontiguous multiple selections. Once you make selections and select OK, the number of selected series and their names are listed in the Series to Process field of the calling window (with ellipses if not all the names will fit).

When invoked from Automatic Model Fitting, the Series to Process window shows all the numeric variables in the input data set except the time id variable. These are the series which are currently available for model fitting.

When invoked from Produce Forecasts, the Series to Process window shows all the series in the input data set for which models have been fit. These are the series which are currently available for forecasting.



AIRRPMD	Airline Revenue Passenger Miles Domestic
AIRRPMT	Airline Revenue Passenger Miles Total
CHEMICAL	Sales: Chemicals and Allied Products
COAL	Bituminous Coal Production
DURABLES	Sales: Durable Goods Industries, Total
HS1FAM	Housing Starts, One-family structures
HSTOTAL	Housing Starts, Total Private
NONDUR	Sales: Nondurable Goods Industries Total
PETROL	Sales: Petroleum and Coal Products
TOBACCO	Sales: Tobacco Products
VEHICLES	Sales: Motor Vehicles and Parts

Controls and Fields

`OK`
: closes the window and applies the series selection(s) which have been made. At least one series must be selected to close the window.

`Cancel`
: closes the window, ignoring series selections which have been made, if any.

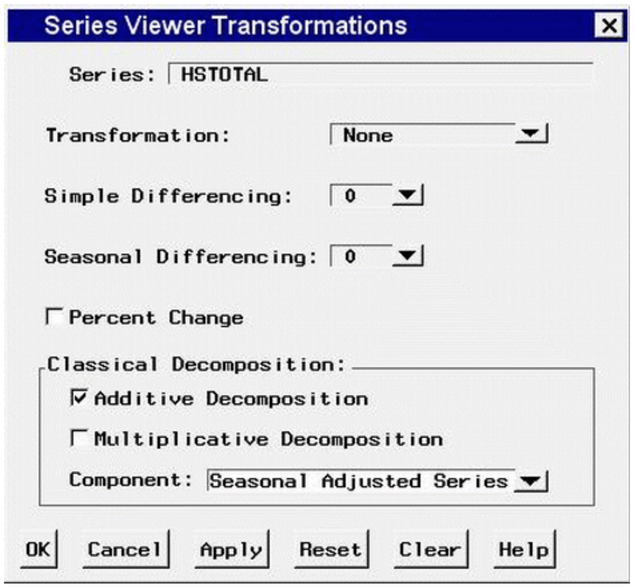
`Clear`
: deselects all series in the selection list.

`All`
: selects all series in the selection list.

Series Viewer Transformations Window

Use the Series Viewer Transformations window to view plots of transformations of the current series in the Time Series Viewer window. It provides a larger set of transformations than those available from the viewer window's tool bar. It is invoked using "Other Transformations" under the Tools pull-down menu of the Time Series Viewer window. The options that you specify in this window are applied to the series displayed in the Time Series Viewer window when you select "OK" or "Apply".

Use the Apply button if you want to make repeated transformations to a series without having to close and reopen the Series Viewer Transformations window each time.


Series Viewer Transformations

Series: HSTOTAL

Transformation: None

Simple Differencing: 0

Seasonal Differencing: 0

☐ Percent Change

Classical Decomposition:

☑ Additive Decomposition

☐ Multiplicative Decomposition

Component: Seasonal Adjusted Series

Controls and Fields

`Series`
: is the variable name for the current time series.

`Transformation`
: is the transformation applied to the time series displayed in the Time Series Viewer window. Select Log, Logistic, Square Root, Box-Cox, or none from the popup list.

`Simple Differencing`
: is the order of differencing applied to the time series displayed in the Time Series Viewer window. Select a number from 0 to 5 from the popup list.

`Seasonal Differencing`
: is the order of seasonal differencing applied to the time series displayed in the Time Series Viewer window. Select a number from 0 to 3 from the popup list.

`Percent Change`
: is a check box that if selected displays the series in terms of percent change from the previous period.

`Additive Decomposition`
: is a check box which produces a display of a selected series component derived using additive decomposition.

`Multiplicative Decomposition`
: is a check box which produces a display of a selected series component derived using multiplicative decomposition.

`Component`
: selects a series component to display when either additive or multiplicative decomposition is turned on. You can display the seasonally adjusted component, the trend-cycle component, the seasonal component, or the irregular component, that is, the residual which remains after removal of the

other components. The heading in the viewer window shows which component is currently displayed.

`OK`
: applies the transformation options you selected to the series displayed in the Time Series Viewer window and closes the Series Viewer Transformations window.

`Cancel`
: closes the Series Viewer Transformations window without changing the series displayed by the Time Series Viewer window.

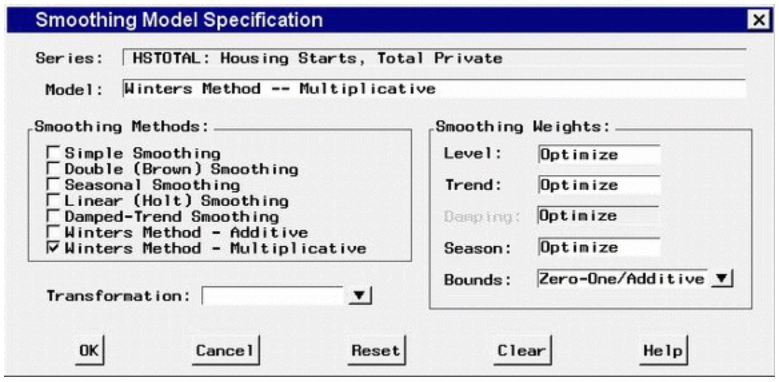
`Apply`
: applies the transformation options you selected to the series displayed in the Time Series Viewer window without closing the Series Viewer Transformations window.

`Reset`
: resets the transformation options to their initial values upon entry to the Series Viewer Transformations window.

`Clear`
: resets the transformation options to their default values (no transformations).

Smoothing Model Specification Window

Use the Smoothing Model Specification window to specify and fit exponential smoothing and Winters method models. Access it from the Develop Models window using the Fit Model submenu of the Edit pull-down or from the pop-up menu when you click an empty area of the model table.



Series: HSTOTAL: Housing Starts, Total Private
Model: Winters Method -- Multiplicative

Smoothing Methods:
- [ ] Simple Smoothing
- [ ] Double (Brown) Smoothing
- [ ] Seasonal Smoothing
- [ ] Linear (Holt) Smoothing
- [ ] Damped-Trend Smoothing
- [ ] Winters Method - Additive
- [✓] Winters Method - Multiplicative

Transformation:

Smoothing Weights:
Level: Optimize
Trend: Optimize
Damping: Optimize
Season: Optimize
Bounds: Zero-One/Additive

Controls and Fields

`Series`
: is the name and variable label of the current series.

`Model`
: is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Smoothing Methods`

`Simple Smoothing`
: specifies simple (single) exponential smoothing.

`Double (Brown) Smoothing`
: specifies double exponential smoothing using Brown's one parameter model (single exponential smoothing applied twice).

`Seasonal Smoothing`
: specifies seasonal exponential smoothing. (This is like Winters method with the trend term omitted.)

`Linear (Holt) Smoothing`
: specifies exponential smoothing of both the series level and trend (Holt's two parameter model).

`Damped-Trend Smoothing`
: specifies exponential smoothing of both the series level and trend with a trend damping weight.

`Winters Method - Additive`
: specifies Winters method with additive seasonal factors.

`Winters Method - Multiplicative`
: specifies Winters method with multiplicative seasonal factors.

`Smoothing Weights`
: displays the values used for the smoothing weights. By default, the Smoothing Weights fields are set to "optimize", which means that the system will compute the weight values that best fit the data. You can also type smoothing weight values in these fields.

`Level`
: is the smoothing weight used for the level of the series.

`Trend`
: is the smoothing weight used for the trend of the series.

`Damping`
: is the smoothing weight used by the damped-trend method to damp the forecasted trend towards zero as the forecast horizon increases.

`Season`
: is the smoothing weight used for the seasonal factors in Winters method and seasonal exponential smoothing.

`Transformation`
: displays the series transformation specified for the model. When a transformation is specified, the model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the model predictions. Select `Log`, `Logistic`, `Square Root`, `Box-Cox`, or `None` from the popup list.

`Bounds`
: displays the constraints imposed on the fitted smoothing weights. Select one of the following from the popup list:

`Zero-One/Additive`
: sets the smoothing weight optimization region to the the intersection of the region bounded by the intervals from zero (0.001) to one (0.999) and the additive invertible region. This is the default.

`Zero-One Boundaries`
: sets the smoothing weight optimization region to the region bounded by the intervals from zero (0.001) to one (0.999).

`Additive Invertible`
: sets the smoothing weight optimization region to the additive invertible region.

`Unrestricted`
: sets the smoothing weight optimization region to be unbounded.

`Custom`
: brings up the `Smoothing Weights` window to enable you to customize the constraints for smoothing weights optimization.

`OK`
: closes the Smoothing Model Specification window and fits the model you specified.

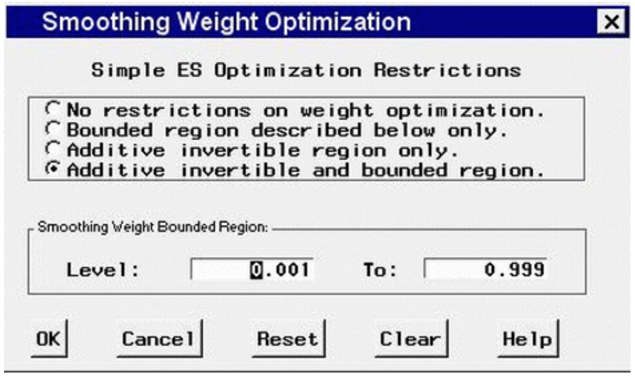
`Cancel`
: closes the Smoothing Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window. This may be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

Smoothing Weight Optimization Window

Use the Smoothing Weight Optimization window to specify constraints for the automatic fitting of smoothing weights for exponential smoothing and Winters method models. Access it from the Smoothing Models Specification window when you select “Custom” in the “Bounds” combo box.



Simple ES Optimization Restrictions

- No restrictions on weight optimization.
- Bounded region described below only.
- Additive invertible region only.
- [✓ Additive invertible and bounded region.]

Smoothing Weight Bounded Region:

Level: 0.001 To: 0.999

Controls and Fields

`No restrictions`
: when selected, specifies unrestricted smoothing weights.

`Bounded region`
: when selected, restricts the fitted smoothing weights to be within the bounds that you specify with the “Smoothing Weight Bounded Region” options.

`Additive invertible region`
: when selected, restricts the fitted smoothing weights to be within the additive invertible region of the parameter space of the ARIMA model equivalent to the smoothing model. (See the section "Smoothing Models" in Chapter 41, “Forecasting Process Details,” for details.)

`Additive invertible and bounded region`
: when selected, restricts the fitted smoothing weights to be both within the additive invertible region and within bounds that you specify.

`Smoothing Weight Bounded Region`
: is a group of numeric entry fields that enable you to specify lower and upper limits on the fitted value of each smoothing weight. The fields that appear in this part of the window depend on the kind of smoothing model that you specified.

`OK`
: closes the window and sets the Bounds options that you specified.

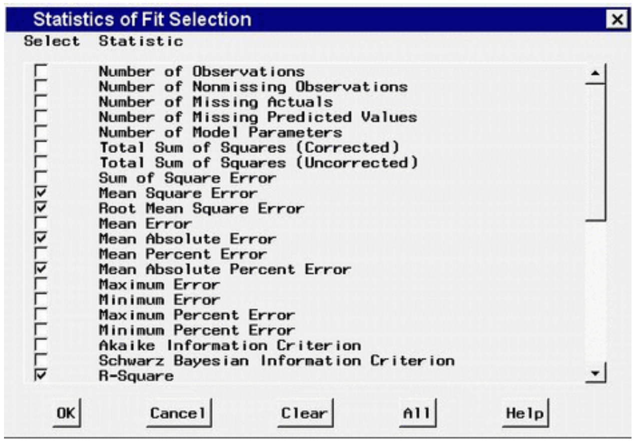
`Cancel`
: closes the window without changing the Bounds option. Any values you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window.

`Clear`
: resets all options to their default values.

Statistics of Fit Selection Window

Use the Statistics of Fit Selection window to specify which of the available goodness-of-fit statistic are reported for models you fit and are available for selection as the model selection criterion used by the automatic selection process. This window is available under the Options pull-down menu in the Develop Models, Automatic Model Fitting, Produce Forecasts, and Model List windows, and from the Statistics button of the Model Fit Comparison window and Automatic Model Fitting results windows.



Select	Statistic
☐	Number of Observations
☐	Number of Nonmissing Observations
☐	Number of Missing Actuals
☐	Number of Missing Predicted Values
☐	Number of Model Parameters
☐	Total Sum of Squares (Corrected)
☐	Total Sum of Squares (Uncorrected)
☐	Sum of Square Error
✓	Mean Square Error
✓	Root Mean Square Error
☐	Mean Error
✓	Mean Absolute Error
☐	Mean Percent Error
✓	Mean Absolute Percent Error
☐	Maximum Error
☐	Minimum Error
☐	Maximum Percent Error
☐	Minimum Percent Error
☐	Akaike Information Criterion
☐	Schwarz Bayesian Information Criterion
✓	R-Square

Controls and Fields

`Available Statistics Table`
: list the available statistics. Select a row of the table to select or deselect the statistic shown in that row.

`OK`
: closes the window and applies the selections made.

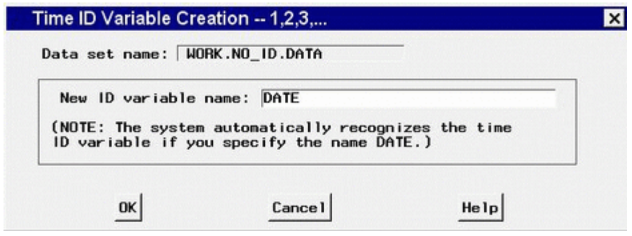
`Cancel`
: closes the window without applying any selections.

`Clear`
: deselects all statistics.

`All`
: selects all statistics.

Time ID Creation – 1,2,3 Window

Use the Time ID Creation – 1,2,3 window to add a time ID variable to an input data set with observation numbers as the ID values. The interval for the series will be 1. Use this approach if the data frequency does not match any of the system's date or date-time intervals, or if other methods of assigning a time ID do not work. To access this window, select "Create from observation numbers" from the the Create popup list in any window where you can select a Time ID variable. For more information, see Chapter 3, "Date Intervals, Formats, and Functions," in this book.



Data set name: WORK.NO_ID.DATA

New ID variable name: DATE

(NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.)

Controls and Fields

`Data set name`
: is the name of the input data set.

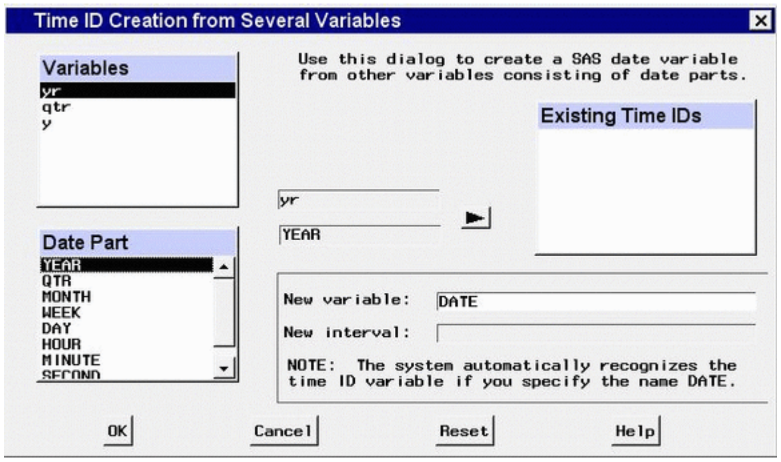
`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without creating a Time ID variable Any options you specified are lost.

Time ID Creation from Several Variables Window

Use the Time ID Creation from Several Variables window to add a SAS date valued time ID variable to an input data set when the input data set already contains several dating variables, such as day, month, and year. To access this window, select "Create from existing variables" from the Create popup list in any window where you can select a Time ID variable. For more information, see "Creating Time ID Variables" in this book.



Use this dialog to create a SAS date variable from other variables consisting of date parts.

yr

YEAR

New variable: DATE

New interval:

NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.

Controls and Fields

`Variables`
: is a list of variables in the input data set. Select existing ID variables from this list.

`Date Part`
: is a list of date parts that you can specify for the selected ID variable. For each ID variable that you select from the Variables list, select the Date Part value that describes what the values of the ID variable represent.

`arrow button`
: Use this button to move the selected existing ID variable and date part specification to the "Existing Time IDs" list. Once you have done this, you can select another ID variable from the Variables list.

`New variable`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`New interval`
: is the time interval between observations in the input data set implied by the date part ID variables you have selected.

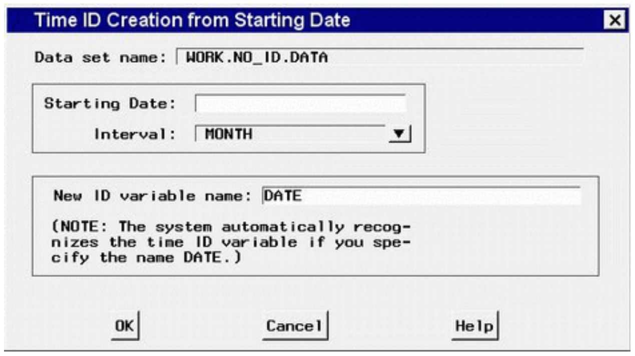
`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without creating a time ID. Any options you specified are lost.

`Reset`
: resets the options to their initial values upon entry to the window.

Time ID Creation from Starting Date Window

Use the Time ID Creation from Starting Date window to add a SAS date valued time ID variable to an input data set. This is a convenient way to add a time ID of any interval as long as you know the starting date of the series. To access this window, select “Create from starting date and frequency” from the Create popup list in any window where you can select a Time ID variable. For more information, see “Creating Time ID Variables” in this book.


Time ID Creation from Starting Date

Data set name: WORK.NO_ID.DATA

Starting Date:

Interval: MONTH

New ID variable name: DATE

(NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.)

Controls and Fields

`Data set name`
: is the name of the input data set.

`Starting Date`
: is the starting date for the time series in the data set. Enter a date value in this field, using a form recognizable by a SAS date informat, for example, 1998:1, feb1997, or 03mar1998.

`Interval`
: is the time interval between observations in the data set. Select an interval from the popup list.

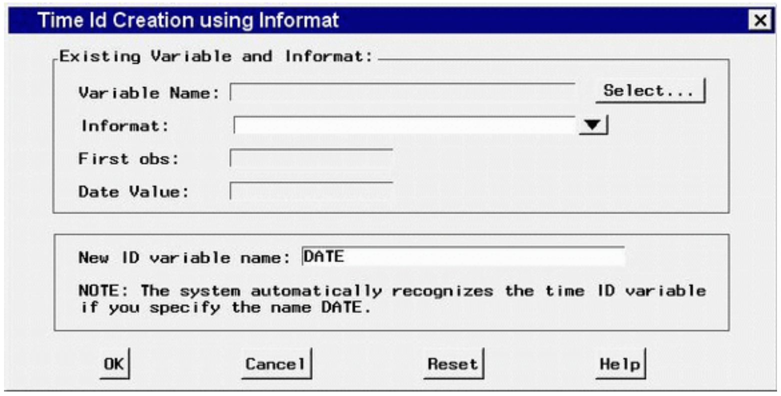
`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without changing the input data set. Any options you specified are lost.

Time ID Creation using Informat Window

Use the Time ID Creation using Informat window to add a SAS date valued time ID variable to an input data set. Use this window if your data set contains a date variable which is stored as a character string. Using the appropriate SAS date informat, the date string is read in and used to create a date or date-time variable. To access this window, select “Create from existing variable/informat” from the Create popup list in any window where you can select a Time ID variable.



Time Id Creation using Informat

Existing Variable and Informat:

Variable Name:

Informat:

First obs:

Date Value:

New ID variable name: DATE

NOTE: The system automatically recognizes the time ID variable if you specify the name DATE.

Controls and Fields

`Variable Name`
: is the name of an existing ID variable in the input data set. Click the Select button to select a variable.

`Select button`
: brings up a list of variables in the input data set for you to select from.

`Informat`
: is a SAS date or datetime informat for reading date or datetime value from the values of the specified existing ID variable. You can type in an informat or select one from the popup list.

`First Obs`
: is the value of the variable you selected from the first observation in the data set, displayed here for convenience.

`Date Value`
: is the SAS date or datetime value read from the first observation value using the informat that you specified.

`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

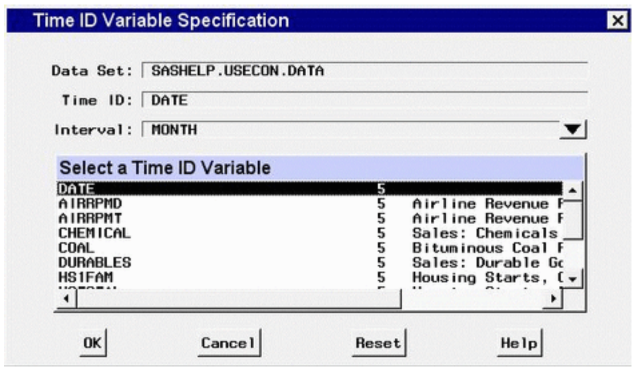
`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without changing the input data set. Any options you specified are lost.

`Reset`
: resets the options to their initial values upon entry to the window.

Time ID Variable Specification Window

Use the Time ID Variable Specification window to specify a variable in the input data set which contains the SAS date or datetime value of each observation. You do not need to use this window if your time ID variable is named `date`, `time`, or `datetime`, since these are picked up automatically. Invoke the window from the Select button to the right of the Time ID field in the Data Set Selection, Automatic Model Fitting, Produce Forecasts, Series Selection, and Time Series Forecasting windows.



Data Set: SASHELP.USECON.DATA
Time ID: DATE
Interval: MONTH

Select a Time ID Variable

```
DATE                    5
AIRRPMD                 5   Airline Revenue F
AIRRPMT                 5   Airline Revenue F
CHEMICAL                5   Sales: Chemicals
COAL                    5   Bituminous Coal F
DURABLES                5   Sales: Durable Gc
HS1FAM                  5   Housing Starts, (
```

Controls and Fields

`Data Set`
: is the name of the current input data set.

`Time ID`
: is the name of the currently selected Time ID variable, if any.

`Interval`
: is the time interval between observations (data frequency) in the input data set.

`Select a Time ID Variable`
: is a selection list of variables in the input set. Select one variable to assign it as the Time ID variable.

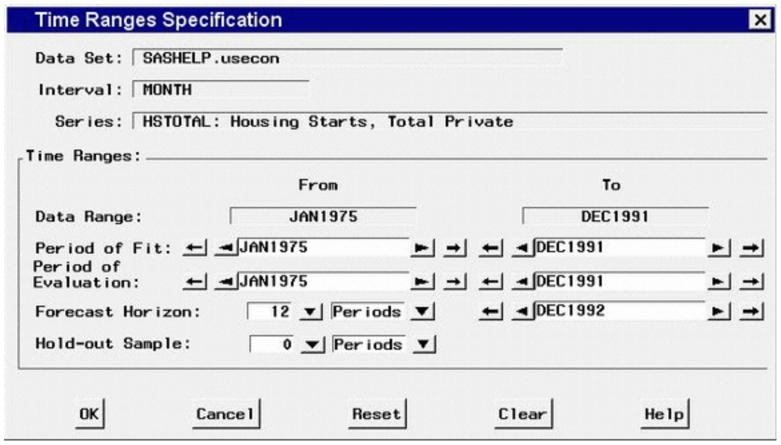
`OK`
: closes the window and retains the selection made, if it is a valid time ID.

`Cancel`
: closes the window and ignores any selection made.

`Reset`
: restores the time ID variable to the one assigned when the window was initially opened, if any.

Time Ranges Specification Window

Use the Time Ranges Specification window to control the period of fit and evaluation and the forecasting horizon. Invoke this window from the Options menu in the Develop Models, Manage Forecasting Project, and Model Viewer windows or the Set Ranges button in the Develop Models window.



Data Set: SASHELP.usecon
Interval: MONTH
Series: HSTOTAL: Housing Starts, Total Private

Time Ranges:

	From		To	
Data Range:	JAN1975		DEC1991	
Period of Fit:	JAN1975		DEC1991	
Period of Evaluation:	JAN1975		DEC1991	
Forecast Horizon:	12	Periods	DEC1992	
Hold-out Sample:	0	Periods		

Controls and Fields

`Data Set`
: is the name of the current input data set.

`Interval`
: is the time interval (data frequency) for the input data set.

`Series`
: is the variable name and label of the current time series.

`Data Range`
: gives the date of the first and last nonmissing data values available for the current series in the input data set.

`Period of Fit`
: gives the starting and ending dates of the period of fit. This is the time range used for estimating model parameters. By default, it is the same as the data range. You can type dates in these fields, or you can use the arrow buttons to the left and right of the date fields to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (Refer to "SAS Language Reference" for information on SAS date informats.) The inner arrows increment by periods, the outer arrows increment by larger amounts, depending on the data interval.

`Period of Evaluation`
: gives the starting and ending dates of the period of evaluation. This is the time range used for evaluating models in terms of statistics of fit. By default, it is the same as the data range. You can type dates in these fields, or you can use the control arrows to the left and right of the date fields to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (Refer to "SAS Language Reference" for information on SAS date informats.) The inner arrows increment by periods, the outer arrows increment by larger amounts, depending on the data interval.

`Forecast Horizon`
: is the forecasting horizon expressed as a number of forecast periods or number of years (or number of weeks for daily data). You can type a number or select one from the popup list. The ending date for the forecast period is automatically updated when you change the number of forecasts periods.

`Forecast Horizon - Units`
: indicates whether the Forecast Horizon value represents periods or years (or weeks for daily data).

`Forecast Horizon Date Value`
: is the date of the last forecast observation. You can type a date in this field, or you can use the arrow buttons to the left and right of the date field to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (Refer to "SAS Language Reference" for information on SAS date informats.) The Forecast Horizon is automatically updated when you change the ending date for the forecast period.

`Hold-out Sample`
: specifies that a number of observations or years (or weeks) of data at the end of the data range are used for the period of evaluation with the remainder of data used as the period of fit. You can type a number in this field or select one from the popup list. When the hold-out sample value is changed, the Period of Fit and Period of Evaluation ranges are changed to reflect the hold-out sample specification.

`Hold-out Sample - Units`
: indicates whether the hold-out sample field represents periods or years (or weeks for daily data).

`OK`
: closes the window and stores the specified changes.

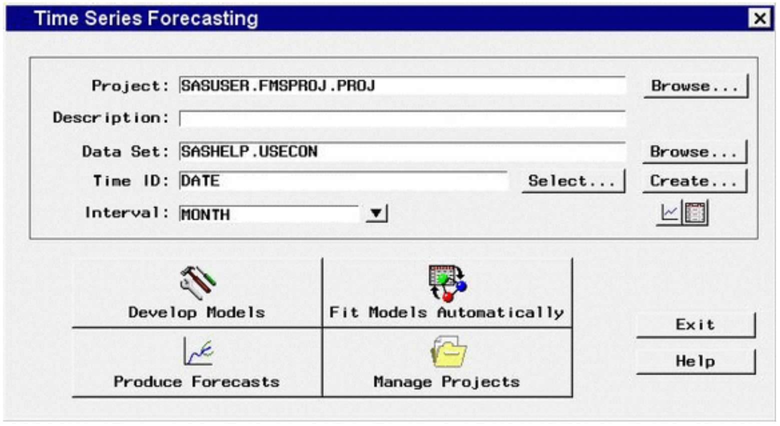
`Cancel`
: closes the window without saving changes. Any options you specified are lost.

`Reset`
: resets the options to their initial values upon entry to the window.

`Clear`
: resets all options to their default values.

Time Series Forecasting Window

The Time Series Forecasting window is the main application window which appears when you invoke the Time Series Forecasting System. It allows you to specify a project file and an input data set and provides access to the other windows described in this chapter.



Project: SASUSER.FMSPROJ.PROJ
Description:
Data Set: SASHELP.USECON
Time ID: DATE
Interval: MONTH

Controls and Fields

`Project`
: is the name of the SAS catalog entry in which forecasting models and other results will be stored and from which previously stored results are loaded into the forecasting system. You can specify the project by typing a SAS catalog entry name in this field or by selecting the Browse button to right of this field. If you specify the name of an existing catalog entry, the information in the project file is loaded. If you specify a one level name, the catalog name is assumed to be `fmsproj` and the library is assumed to be `sasuser`. For example, `samproj` is equivalent to `sasuser.fmsproj.samproj`.

Project `Browse` button
: brings up the Forecasting Project File Selection window to enable you to select and load the project from a list of previously stored projects.

`Description`
: is a descriptive label for the forecasting project. The description you type in this field will be stored with the catalog entry shown in the Project field.

`Data Set`
: is the name of the current input data set. To specify the input data set, you can type the data set name in this field or use the Browse button to the right of the field.

Data set `Browse` button
: brings up the Data Set Selection window to enable you to select the input data set.

`Time ID`
: is the name of the ID variable for the input data set. To specify the ID variable, you can type the ID variable name in this field or use the Select

button. If the time ID variable is named `date`, `time`, or `datetime`, it is automatically picked up by the system.

`Select` button
: brings up the Time ID Variable Specification window.

`Create` button
: brings up a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the current input data set. If the interval is not automatically filled in, you can type an interval name or select one from the popup list. For more information on intervals, see "Time Series Data Sets, ID variables, and Time Intervals" in this book.

`View Series Graphically` icon
: brings up the Time Series Viewer window to display plots of series in the current input data set.

`View Data as a Table`
: brings up a Viewtable window for browsing the selected input data set.

`Develop Models`
: brings up the Develop Models window to enable you to fit forecasting models to individual time series and choose the best models to use to produce the final forecasts of each series.

`Fit Models Automatically`
: brings up the Automatic Model Fitting window for applying the automatic model selection process to all series or to selected series in an input data set.

`Produce Forecast`
: brings up the Produce Forecasts window for producing forecasts for the series in the current input data set for which you have fit forecasting models.

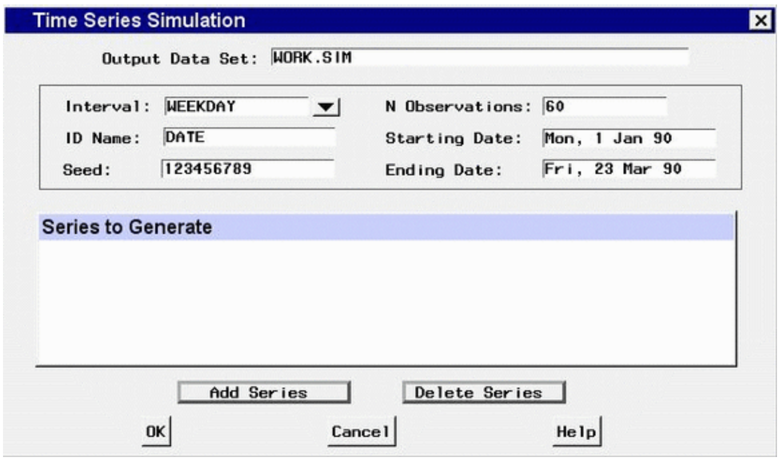
`Manage Projects`
: brings up the Manage Forecasting Project window for viewing or editing information stored in projects.

`Exit`
: closes the Time Series Forecasting system.

`Help`
: accesses the help system.

Time Series Simulation Window

Use the Time Series Simulation window to create a data set of simulated series generated by ARIMA processes. Access this window from the Tools pull-down menu in the Develop Models and Manage Forecasting Project windows.


Time Series Simulation

Output Data Set: WORK.SIM

Interval: WEEKDAY
ID Name: DATE
Seed: 123456789
N Observations: 60
Starting Date: Mon, 1 Jan 90
Ending Date: Fri, 23 Mar 90

Controls and Fields

`Output Data Set`
: is the name of the data set to be created. Type in a one or two level SAS data set name.

`Interval`
: is the time interval between observations (data frequency) in the simulated data set. Type in an interval name or select one from the popup list.

`Seed`
: is the seed for the random number generator used to produce the simulated time series.

`N Observations`
: is the number of time periods to simulate.

`Starting Date`
: is the starting date for the simulated observations. Type in a date in a form recognizable by a SAS data informat, for example, 1998:1, feb1997, or 03mar1998.

`Ending Date`
: is the ending date for the simulated observations. Type in a date in a form recognizable by a SAS data informat.

`Series to Generate`
: is the list of variable names and ARIMA processes to simulate.

`Add Series`
: brings up the ARIMA Process Specification window to enable you to add entries to the Series to Generate list.

`Delete Series`
: deletes selected (highlighted) entries from the Series to Generate list.

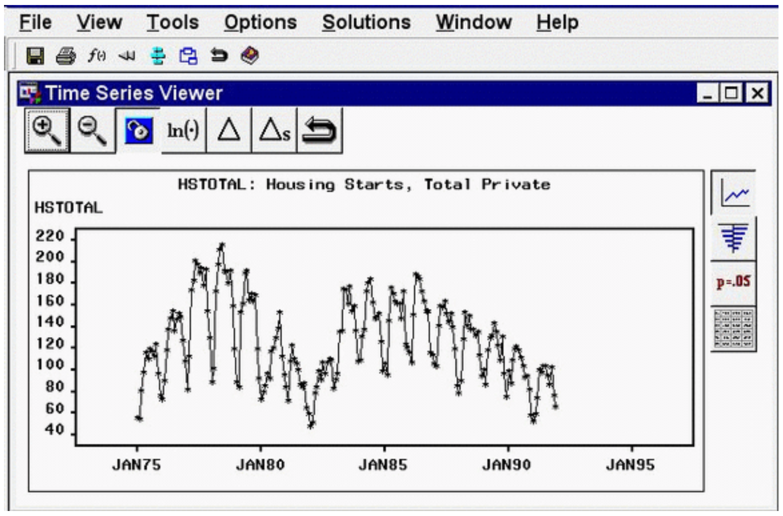
`OK`
: closes the Time Series Simulation window and performs the specified simulations and creates the specified data set.

`Cancel`
closes the window without creating a simulated data set. Any options you specified are lost.

Time Series Viewer Window

Use the Time Series Viewer window to explore time series data using plots, transformations, statistical tests, and tables. It is available as a stand-alone application and as part of the Time Series Forecasting System. To use it as a stand-alone application, select it from the Analysis submenu of the Solutions pull-down menu, or use the `tsview` command (see Chapter 39, "Command Reference," in this book). To use it within the Time Series Forecasting System, select the View Series Graphically icon in the Time Series Forecasting, Develop Models, or Model List window, or select "Series" from the View pull-down menu of the Develop Models, Manage Project, or Model List window.

The various plots and tables available are referred to as *views*. The following "View Selection Icons" section explains how to change the view.



The state of the Time Series Viewer window is controlled by the current series, the current series transformation specification, and the currently selected view. You can resize this window, and you can use other windows without closing the Time Series Viewer window. You can explore a number of series conveniently by keeping the Series Selection window open. Each time you make a selection, the viewer window is updated to show the selected series. Keep both windows visible, or switch between them using the Next Viewer tool bar icon or the F12 function key.

You can bring up multiple Time Series Viewer windows. This enables you to "freeze" a plot so you can come back to it later, or compare two plots side by side on your screen. To do this, unlink the viewer using the Link/Unlink icon on the window's tool bar or the corresponding item in the Tools pull-down menu. While the viewer window remains unlinked, it is not updated when other selections are made in the Series Selection window. Instead, when you select a series and click the Graph button, a new Time Series Viewer window is invoked. You can continue this process to bring up as many viewer windows as you want. The Next Viewer icon and corresponding F12 function key are useful for navigating between windows when they are not simultaneously visible on your screen.

A wide range of series transformations is available. Basic transformations are available from the window’s horizontal tool bar, and others are available by selecting “Other Transformations” from the Tools pull-down menu.

Horizontal Tool Bar

The Time Series Viewer window contains a horizontal tool bar with the following icons:

`Zoom in`
: changes the mouse cursor into cross hairs that you can use with the left mouse button to drag out a region of the time series plot to zoom in on. In the Autocorrelations view and the White Noise and Stationarity Tests view, Zoom In reduces the number of lags displayed.

`Zoom out`
: reverses the previous Zoom In action and expands the time range of the plot to show more of the series. In the Autocorrelations view and the White Noise and Stationarity Tests view, Zoom Out increases the number of lags displayed.

`Link/Unlink viewer`
: disconnects or connects the Time Series Viewer window to the window in which the series was selected. When the Viewer is linked, it always shows the current series. If you select another series, linked Viewers are updated. Unlinking a Viewer freezes its current state, and changing the current series has no effect on the Viewer’s display. The View Series action creates a new Series Viewer window if there is no linked Viewer. By using the unlink feature, you can bring up several Time Series Viewer windows and display several different series simultaneously.

`Log Transform`
: applies a log transform to the current view. This can be combined with other transformations: The current transformations are shown in the title.

`Difference`
: applies a simple difference to the current view. This can be combined with other transformations: The current transformations are shown in the title.

`Seasonal Difference`
: applies a seasonal difference to the current view. For example, if the data are monthly, the seasonal cycle is one year. Each value has subtracted from it the value from one year previous. This can be combined with other transformations: The current transformations are shown in the title.

`Close`
: closes the Time Series Viewer window and returns to the window from which it was invoked.

Vertical Toolbar View Selection Icons

At the right-hand side of the Time Series Viewer window is a vertical tool bar used to select the kind of plot or table that the Viewer displays.

`Series`
: displays a plot of series values over time.

`Autocorrelations`
: displays plots of the sample autocorrelations, partial autocorrelation, and inverse autocorrelation functions for the series, with lines overlaid at plus and minus two standard errors.

`White Noise and Stationarity Tests`
: displays horizontal bar charts representing results of white noise and stationarity tests. The first bar chart shows the significance probability of the Ljung-Box chi-square statistic computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the series is white noise. Click on any of the bars to display an interpretation.

  The second bar chart shows tests of stationarity, where longer bars favor the conclusion that the series is stationary. Each bar displays the significance probability of the augmented Dickey-Fuller unit root test to the given autoregressive lag. Long bars represent higher levels of significance against the null hypothesis that the series contains a unit root. For seasonal data, a third bar chart appears for seasonal root tests. Click on any of the bars to display an interpretation.

`Data Table`
: displays a data table containing the values in the input data set.

Menu Bar

`File`

  `Save Graph`
  : saves the current plot as a SAS/GRAPH grseg catalog entry in a default or most recently specified catalog. This item is grayed out in the Data Table view.

  `Save Graph as`
  : saves the current graph as a SAS/GRAPH grseg catalog entry in a SAS catalog that you specify and/or as an Output Delivery System (ODS) object. By default, an html page is created, with the graph embedded as a gif image. This item is grayed out in the Data Table view.

  `Save Data`
  : saves the data displayed in the viewer window to an output SAS data set. This item is grayed out in the Series view.

  `Save Data as`
  : saves the data in a SAS data set that you specify and/or as an Output Delivery System (ODS) object. By default, an html page is created, with the data displayed as a table.

  `Import Data`
  : is available if you license SAS/Access software. It brings up an

Import Wizard which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It brings up an Export Wizard which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Graph`
: prints the plot displayed in the viewer window. This item is grayed out in the Data Table view.

`Print Data`
: prints the data displayed in the viewer window. This item is grayed out in the Series view.

`Print Setup`
: brings up the Print Setup window, which allows you to access your operating system print setup.

`Print Preview`
: brings up a preview window to show how your plots will look when printed.

`Close`
: closes the Time Series Viewer window and returns to the window from which it was invoked.

`View`

`Series`
: displays a plot of series values over time. This is the same as the Series icon in the vertical tool bar.

`Autocorrelations`
: displays plots of the sample autocorrelation, partial autocorrelation, and inverse autocorrelation functions for the series. This is the same as the Autocorrelations icon in the vertical tool bar.

`White Noise and Stationarity Tests`
: displays horizontal bar charts representing results of white noise and stationarity tests. This is the same as the White Noise and Stationarity Tests icon in the vertical tool bar.

`Data Table`
: displays a data table containing the values in the input data set. This is the same as the Data Table icon in the vertical tool bar.

`Zoom In`
: zooms the display. This is the same as the Zoom In icon in the window's horizontal tool bar.

`Zoom Out`
: undoes the last zoom in action. This is the same as the Zoom Out icon in the window's horizontal tool bar.

`Zoom Way Out`
: reverses all previous Zoom In actions and expands the time range of the plot to show all of the series, or shows the maximum number of

lags in the Autocorrelations View or the White Noise and Stationarity Tests view.

`Tools`

`Log Transform`
: applies a log transformation. This is the same as the Log Transform icon in the window's horizontal tool bar.

`Difference`
: applies simple differencing. This is the same as the Difference icon in the window's horizontal tool bar.

`Seasonal Difference`
: applies seasonal differencing. This is the same as the Seasonal Difference icon in the window's horizontal tool bar.

`Other Transformations`
: brings up the Series Viewer Transformations window to enable you to apply a wide range of transformations.

`Diagnose Series`
: brings up the Series Diagnostics window to determine the kinds of forecasting models appropriate for the current series.

`Define Interventions`
: brings up the Interventions for Series window to enable you to edit or add intervention effects for use in modeling the current series.

`Link Viewer`
: connects or disconnects the Time Series Viewer window to the window from which series are selected. This is the same as the Link item in the window's horizontal tool bar.

`Options`

`Number of Lags`
: brings up a window to let you specify the number of lags shown in the Autocorrelations view and the White Noise and Stationarity Tests view. You can also use the Zoom In and Zoom Out actions to control the number of lags displayed.

`Correlation Probabilities`
: controls whether the bar charts in the Autocorrelations view represent significance probabilities or values of the correlation coefficient. A check mark or filled check box next to this item indicates that significance probabilities are displayed. In each case the bar graph horizontal axis label changes accordingly.

Mouse Button Actions

You can examine the data value and date of individual points in the Series view by positioning the mouse cursor over the point and clicking the left mouse button. The date and value are displayed in a box that appears in the upper right corner of the Viewer window. Click the mouse elsewhere or select any action to dismiss the data box.

You can examine the values of the bars and confidence limits at different lags in the Autocorrelations view by clicking on individual bars in the vertical bar charts.

You can display an interpretation of the tests in the White Noise and Stationarity Tests view by clicking on the bars.

When you select the Zoom In action, you can use the mouse to define a region of the graph to take a closer look at. Position the mouse cursor at one corner of the region, press the left mouse button, and move the mouse cursor to the opposite corner of the region while holding the left mouse button down. When you release the mouse button, the plot is redrawn to show an expanded view of the data within the region you selected.

Chapter 41
Forecasting Details

Chapter Contents

Chapter 41
Forecasting Process Details

This chapter provides computational details on several aspects of the Time Series Forecasting System.

Forecasting Process Summary

This section summarizes the forecasting process.

Parameter Estimation

The parameter estimation process for ARIMA and smoothing models is described graphically in Figure 41.1.

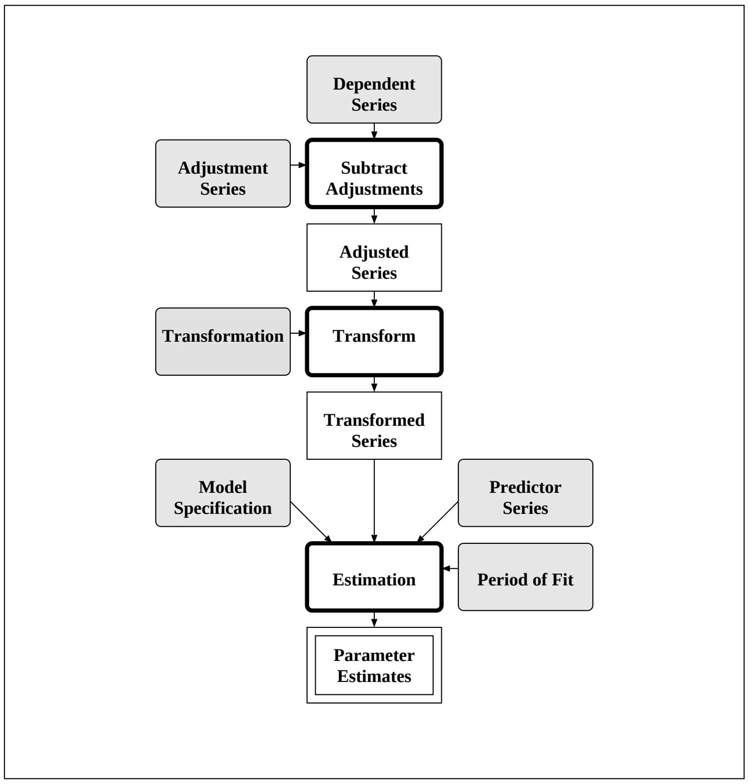


Figure 41.1. Model Fitting Flow Diagram

The specification of smoothing and ARIMA models is described in Chapter 36, “Specifying Forecasting Models.” Computational details for these kinds of models are provided in the following sections "Smoothing Models" and "ARIMA Models." The results of the parameter estimation process are displayed in the Parameter Estimates table of the Model Viewer window along with the estimate of the model variance and the final smoothing state.

Model Evaluation

The model evaluation process is described graphically in Figure 41.2.

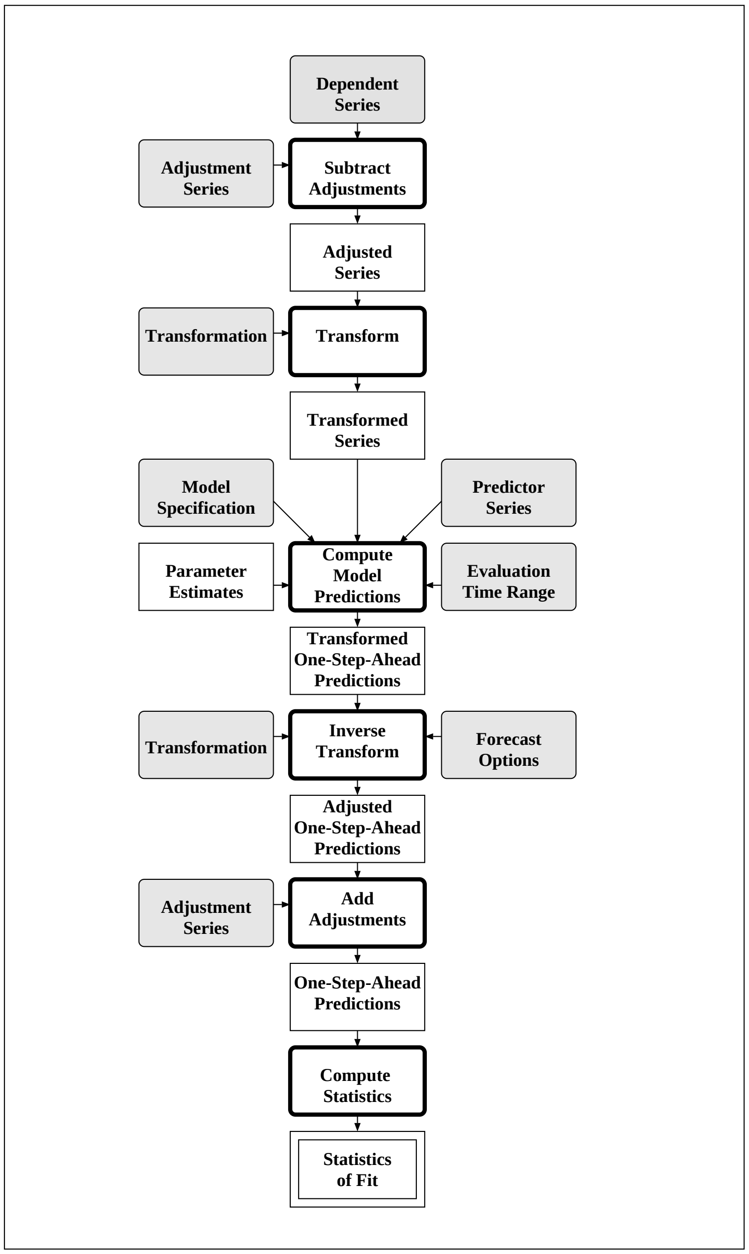


Figure 41.2. Model Evaluation Flow Diagram

Model evaluation is based on the one-step-ahead prediction errors for observations within the period of evaluation. The one-step-ahead predictions are generated from the model specification and parameter estimates. The predictions are inverse transformed (median or mean) and adjustments are removed. The prediction errors (the difference of the dependent series and the predictions) are used to compute the statistics of fit, which are described in the following section "Diagnostic Tests and Statistics of Fit." The results generated by the evaluation process are displayed in the Statistics of Fit table of the Model Viewer window.

Forecasting

The forecasting generation process is described graphically in Figure 41.3.

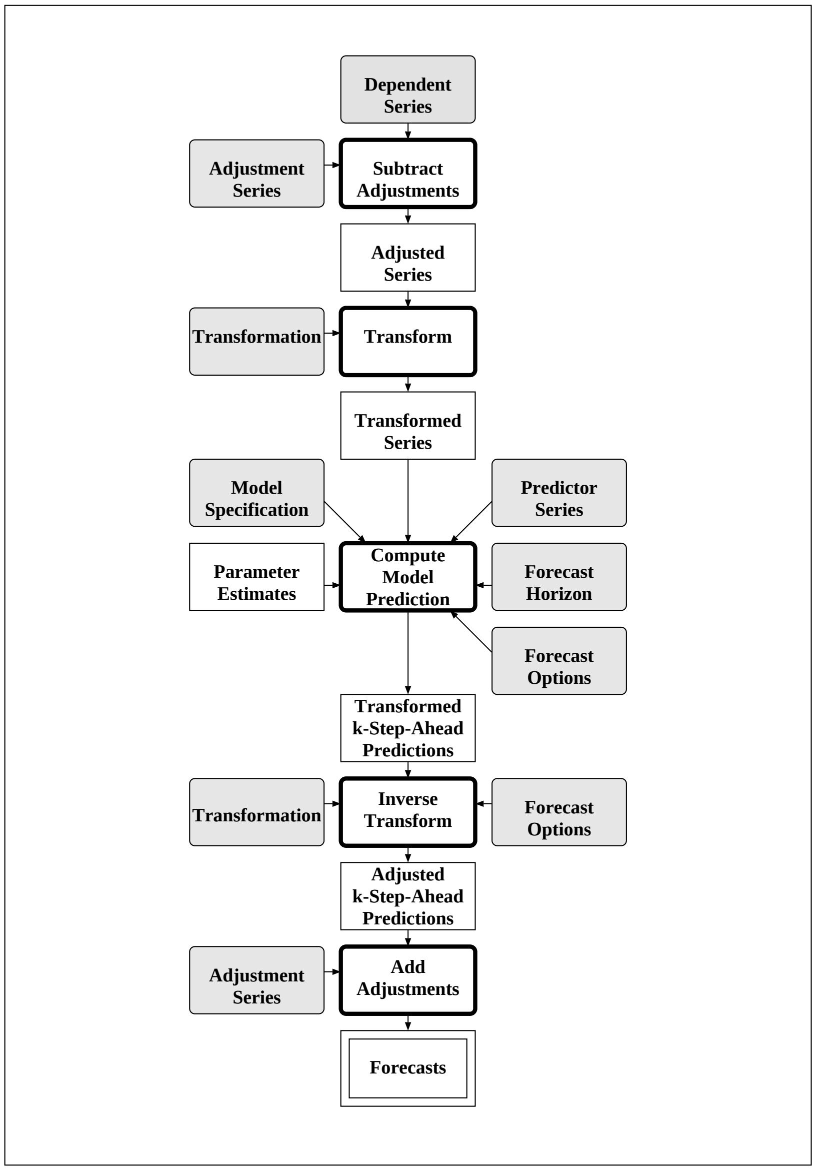


Figure 41.3. Forecasting Flow Diagram

The forecasting process is similar to the model evaluation process described in the preceding section, except that *k*-step-ahead predictions are made from the end of the data through the specified forecast horizon, and prediction standard errors and confidence limits are calculated. The forecasts and confidence limits are displayed in the Forecast plot or table of the Model Viewer window.

Forecast Combination Models

This section discusses the computation of predicted values and confidence limits for forecast combination models. See Chapter 36, "Specifying Forecasting Models," for information on how to specify forecast combination models and their combining weights.

Given the response time series $\{y_t : 1 \leq t \leq n\}$ with previously generated forecasts for the *m* component models, a combined forecast is created from the component forecasts as follows:

Predictions:	$\hat{y}_t = \sum_{i=1}^{m} w_i \hat{y}_{i,t}$
Prediction Errors:	$\hat{e}_t = y_t - \hat{y}_t$

where $\hat{y}_{i,t}$ are the forecasts of the component models and w_i are the combining weights.

The estimate of the root mean square prediction error and forecast confidence limits for the combined forecast are computed by assuming independence of the prediction errors of the component forecasts, as follows:

Standard Errors:	$\hat{\sigma}_t = \sqrt{\sum_{i=1}^{m} w_i^2 \hat{\sigma}_{i,t}^2}$
Confidence Limits:	$\pm \hat{\sigma}_t \mathrm{Z}_{\alpha/2}$

where $\hat{\sigma}_{i,t}$ are the estimated root mean square prediction errors for the component models, α is the confidence limit width, $1 - \alpha$ is the confidence level, and $\mathrm{Z}_{\alpha/2}$ is the $\frac{\alpha}{2}$ quantile of the standard normal distribution.

Since, in practice, there may be positive correlation between the prediction errors of the component forecasts, these confidence limits may be too narrow.

External or User-Supplied Forecasts

This section discusses the computation of predicted values and confidence limits for external forecast models.

Given a response time series y_t and external forecast series $\hat{y}_t$, the prediction errors are computed as $\hat{e}_t = y_t - \hat{y}_t$ for those *t* for which both y_t and $\hat{y}_t$ are nonmissing. The mean square error (MSE) is computed from the prediction errors.

The variance of the *k*-step-ahead prediction errors is set to *k* times the MSE. From these variances, the standard errors and confidence limits are computed in the usual way. If the supplied predictions contain so many missing values within the time range of the response series that the MSE estimate cannot be computed, the confidence limits, standard errors, and statistics of fit are set to missing.

Adjustments

Adjustment predictors are subtracted from the response time series prior to model parameter estimation, evaluation, and forecasting. After the predictions of the adjusted response time series are obtained from the forecasting model, the adjustments are added back to produce the forecasts.

If y_t is the response time series and $X_{i,t}$, $1 \leq i \leq m$ are *m* adjustment predictor series, then the adjusted response series w_t is

$$w_t = y_t - \sum_{i=1}^{m} X_{i,t}$$

Parameter estimation for the model is performed using the adjusted response time series w_t. The forecasts $\hat{w}_t$ of w_t are adjusted to obtain the forecasts $\hat{y}_t$ of y_t.

$$\hat{y}_t = \hat{w}_t + \sum_{i=1}^{m} X_{i,t}$$

Missing values in an adjustment series are ignored in these computations.

Series Transformations

For pure ARIMA models, transforming the response time series may aid in obtaining stationary noise series. For general ARIMA models with inputs, transforming the response time series or one or more of the input time series may provide a better model fit. Similarly, the fit of smoothing models may improve when the response series is transformed.

There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log — is the logarithmic transformation.

$$w_t = \ln(\mathrm{y_t})$$

Logistic — is the logistic transformation.

$$w_t = \ln(\mathrm{cy_t}/(1 - \mathrm{cy_t}))$$

where the scaling factor c is

$$c = (1 - 10^{-6})10^{-\mathrm{ceil}(\log_{10}(\max(\mathrm{y_t})))}$$

and $\mathrm{ceil}(\mathrm{x})$ is the smallest integer greater than or equal to *x*.

Square Root is the square root transformation.

$$w_t = \sqrt{y_t}$$

Box Cox is the Box-Cox transformation.

$$w_t = \begin{cases} \frac{y_t^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \ln(\mathrm{y_t}), & \lambda = 0 \end{cases}$$

Parameter estimation is performed using the transformed series. The transformed model predictions and confidence limits are then obtained from the transformed time-series and these parameter estimates.

The transformed model predictions $\hat{w}_t$ are used to obtain either the minimum mean absolute error (MMAE) or minimum mean squared error (MMSE) predictions $\hat{y}_t$, depending on the setting of the forecast options. The model is then evaluated based on the residuals of the original time series and these predictions. The transformed model confidence limits are inverse-transformed to obtain the forecast confidence limits.

Predictions for Transformed Models

Since the transformations described in the previous section are monotonic, applying the inverse-transformation to the transformed model predictions results in the *median* of the conditional probability density function at each point in time. This is the minimum mean absolute error (MMAE) prediction.

If $w_t = \mathrm{F(y_t)}$ is the transform with inverse-transform $y_t = \mathrm{F^{-1}(w_t)}$, then

$$\mathrm{median(\hat{y}_t) = F^{-1}(E\left[w_t\right]) = F^{-1}(\hat{w}_t)}$$

The minimum mean squared error (MMSE) predictions are the *mean* of the conditional probability density function at each point in time. Assuming that the prediction errors are normally distributed with variance σ_t^2, the MMSE predictions for each of the transformations are as follows:

Log is the conditional expectation of inverse-logarithmic transformation.

$$\hat{y}_t = E[e^{w_t}] = \exp\left(\mathrm{\hat{w}_t} + \sigma_\mathrm{t}^2/2\right)$$

Logistic is the conditional expectation of inverse-logistic transformation.

$$\hat{y}_t = E\left[\frac{1}{c(1+exp(-w_t))}\right]$$

where the scaling factor $c = (1 - e^{-6})10^{-\mathrm{ceil}(\log_{10}(\max(\mathrm{y_t})))}$.

Square Root is the conditional expectation of the inverse-square root transformation.

$$\hat{y}_t = E\left[w_t^2\right] = \hat{w}_t^2 + \sigma_t^2$$

Box Cox is the conditional expectation of the inverse Box-Cox transformation.

$$\hat{y}_t = \begin{cases} E\left[(\lambda w_t + 1)^{1/\lambda}\right], & \lambda{\neq}0 \\ E\left[e^{w_t}\right] = \exp(\hat{\mathrm{w}}_{\mathrm{t}} + \frac{1}{2}\sigma_{\mathrm{t}}^2), & \lambda = 0 \end{cases}$$

The expectations of the inverse logistic and Box-Cox ($\lambda{\neq}0$) transformations do not generally have explicit solutions and are computed using numerical integration.

Smoothing Models

This section details the computations performed for the exponential smoothing and Winters method forecasting models.

Smoothing Model Calculations

The descriptions and properties of various smoothing methods can be found in Gardner (1985), Chatfield (1978), and Bowerman and O'Connell (1979). The following section summarizes the smoothing model computations.

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the underlying model assumed by the smoothing models has the following (additive seasonal) form:

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

where

μ_t	represents the time-varying mean term.
β_t	represents the time-varying slope.
$s_p(t)$	represents the time-varying seasonal contribution for one of the p seasons
ϵ_t	are disturbances.

For smoothing models without trend terms, $\beta_t = 0$; and for smoothing models without seasonal terms, $s_p(t) = 0$. Each smoothing model is described in the following sections.

At each time t, the smoothing models estimate the time-varying components described above with the *smoothing state*. After initialization, the smoothing state is updated for each observation using the *smoothing equations*. The smoothing state at the last nonmissing observation is used for predictions.

Smoothing State and Smoothing Equations

Depending on the smoothing model, the *smoothing state* at time t will consist of the following:

L_t is a smoothed level that estimates μ_t.

T_t is a smoothed trend that estimates β_t.

S_{t-j}, $j = 0, \ldots, p-1$, are seasonal factors that estimate $s_p(t)$.

The smoothing process starts with an initial estimate of the smoothing state, which is subsequently updated for each observation using the *smoothing equations*.

The smoothing equations determine how the smoothing state changes as time progresses. Knowledge of the smoothing state at time $t-1$ and that of the time-series value at time t uniquely determine the smoothing state at time t. The *smoothing weights* determine the contribution of the previous smoothing state to the current smoothing state. The smoothing equations for each smoothing model are listed in the following sections.

Smoothing State Initialization

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the smoothing process first computes the smoothing state for time $t = 1$. However, this computation requires an initial estimate of the smoothing state at time $t = 0$, even though no data exists at or before time $t = 0$.

An appropriate choice for the initial smoothing state is made by backcasting from time $t = n$ to $t = 1$ to obtain a prediction at $t = 0$. The initialization for the backcast is obtained by regression with constant and linear terms and seasonal dummies (additive or multiplicative) as appropriate for the smoothing model. For models with linear or seasonal terms, the estimates obtained by the regression are used for initial smoothed trend and seasonal factors; however, the initial smoothed level for backcasting is always set to the last observation, Y_n.

The smoothing state at time $t = 0$ obtained from the backcast is used to initialize the smoothing process from time $t = 1$ to $t = n$ (refer to Chatfield and Yar 1988).

For models with seasonal terms, the smoothing state is normalized so that the seasonal factors S_{t-j} for $j = 0, \ldots, p-1$ sum to zero for models that assume additive seasonality and average to one for models (such as Winters method) that assume multiplicative seasonality.

Missing Values

When a missing value is encountered at time t, the smoothed values are updated using the *error-correction form* of the smoothing equations with the one-step-ahead prediction error, e_t, set to zero. The missing value is estimated using the one-step-ahead prediction at time $t-1$, that is $\hat{Y}_{t-1}(1)$ (refer to Aldrin 1989). The error-correction forms of each of the smoothing models are listed in the following sections.

Predictions and Prediction Errors

Predictions are made based on the last known smoothing state. Predictions made at time t for k steps ahead are denoted $\hat{Y}_t(k)$ and the associated prediction errors are denoted $e_t(k) = Y_{t+k} - \hat{Y}_t(k)$. The *prediction equation* for each smoothing model is listed in the following sections.

The *one-step-ahead predictions* refer to predictions made at time $t - 1$ for one time unit into the future, that is, $\hat{Y}_{t-1}(1)$, and the *one-step-ahead prediction errors* are more simply denoted $e_t = e_{t-1}(1) = Y_t - \hat{Y}_{t-1}(1)$. The one-step-ahead prediction errors are also the model residuals, and the sum of squares of the one-step-ahead prediction errors is the objective function used in smoothing weight optimization.

The *variance of the prediction errors* are used to calculate the confidence limits (refer to Sweet 1985, McKenzie 1986, Yar and Chatfield 1990, and Chatfield and Yar 1991). The equations for the variance of the prediction errors for each smoothing model are listed in the following sections.

Note: $var(\epsilon_t)$ is estimated by the mean square of the one-step-ahead prediction errors.

Smoothing Weights

Depending on the smoothing model, the smoothing weights consist of the following:

α	is a level smoothing weight.
γ	is a trend smoothing weight.
δ	is a seasonal smoothing weight.
ϕ	is a trend damping weight.

Larger smoothing weights (less damping) permit the more recent data to have a greater influence on the predictions. Smaller smoothing weights (more damping) give less weight to recent data.

Specifying the Smoothing Weights

Typically the smoothing weights are chosen to be from zero to one. (This is intuitive because the weights associated with the past smoothing state and the value of current observation would normally sum to one.) However, each smoothing model (except Winters Method – Multiplicative Version) has an ARIMA equivalent. Weights chosen to be within the ARIMA additive-invertible region will guarantee stable predictions (refer to Archibald 1990 and Gardner 1985). The ARIMA equivalent and the additive-invertible region for each smoothing model are listed in the following sections.

Optimizing the Smoothing Weights

Smoothing weights are determined so as to minimize the sum of squared one-step-ahead prediction errors. The optimization is initialized by choosing from a predetermined grid the initial smoothing weights that result in the smallest sum of squared, one-step-ahead prediction errors. The optimization process is highly dependent on this initialization. It is possible that the optimization process will fail due to the inability to obtain stable initial values for the smoothing weights (refer to Greene 1993 and Judge et al 1980), and it is possible for the optimization to result in a local minima.

The optimization process can result in weights to be chosen outside both the zero-to-one range and the ARIMA additive-invertible region. By restricting weight optimization to additive-invertible region, you can obtain a local minimum with stable predictions. Likewise, weight optimization can be restricted to the zero-to-one range or other ranges. It is also possible to fix certain weights to a specific value and optimize the remaining weights.

Standard Errors

The standard errors associated with the smoothing weights are calculated from the Hessian matrix of the sum of squared, one-step-ahead prediction errors with respect to the smoothing weights used in the optimization process.

Weights Near Zero or One

Sometimes the optimization process results in weights near zero or one.

For Simple or Double (Brown) Exponential Smoothing, a level weight near zero implies that simple differencing of the time series may be appropriate.

For Linear (Holt) Exponential Smoothing, a level weight near zero implies that the smoothed trend is constant and that an ARIMA model with deterministic trend may be a more appropriate model.

For Damped-Trend Linear Exponential Smoothing, a damping weight near one implies that Linear (Holt) Exponential Smoothing may be a more appropriate model.

For Winters Method and Seasonal Exponential Smoothing, a seasonal weight near one implies that a nonseasonal model may be more appropriate and a seasonal weight near zero implies that deterministic seasonal factors may be present.

Equations for the Smoothing Models

Simple Exponential Smoothing

The model equation for simple exponential smoothing is

$$Y_t = \mu_t + \epsilon_t$$

The smoothing equation is

$$L_t = \alpha Y_t + (1 - \alpha) L_{t-1}$$

The error-correction form of the smoothing equation is

$$L_t = L_{t-1} + \alpha e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t$$

The ARIMA model equivalency to simple exponential smoothing is the ARIMA(0,1,1) model

$$(1 - B)Y_t = (1 - \theta B)\epsilon_t$$

$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \alpha \epsilon_{t-j}$$

For simple exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t) \left[1 + \sum_{j=1}^{k-1} \alpha^2 \right] = var(\epsilon_t)(1 + (k-1)\alpha^2)$$

Double (Brown) Exponential Smoothing

The model equation for double exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1 - \alpha)L_{t-1}$$

$$T_t = \alpha(L_t - L_{t-1}) + (1 - \alpha)T_{t-1}$$

This method may be equivalently described in terms of two successive applications of simple exponential smoothing:

$$S_t^{[1]} = \alpha Y_t + (1 - \alpha)S_{t-1}^{[1]}$$

$$S_t^{[2]} = \alpha S_t^{[1]} + (1-\alpha) S_{t-1}^{[2]}$$

where $S_t^{[1]}$ are the smoothed values of Y_t, and $S_t^{[2]}$ are the smoothed values of $S_t^{[1]}$. The prediction equation then takes the form:

$$\hat{Y}_t(k) = (2 + \alpha k/(1-\alpha)) S_t^{[1]} - (1 + \alpha k/(1-\alpha)) S_t^{[2]}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha^2 e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t + ((k-1) + 1/\alpha) T_t$$

The ARIMA model equivalency to double exponential smoothing is the ARIMA(0,2,2) model

$$(1-B)^2 Y_t = (1-\theta B)^2 \epsilon_t$$

$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (2\alpha + (j-1)\alpha^2) \epsilon_{t-j}$$

For double exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t) \left[1 + \sum_{j=1}^{k-1} (2\alpha + (j-1)\alpha^2)^2 \right]$$

Linear (Holt) Exponential Smoothing

The model equation for linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1 - \gamma)T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha\gamma e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t$$

The ARIMA model equivalency to linear exponential smoothing is the ARIMA(0,2,2) model

$$(1 - B)^2 Y_t = (1 - \theta_1 B - \theta_2 B^2)\epsilon_t$$

$$\theta_1 = 2 - \alpha - \alpha\gamma$$

$$\theta_2 = \alpha - 1$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (\alpha + j\alpha\gamma)\epsilon_{t-j}$$

For linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

$$\{0 < \gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (\alpha + j\alpha\gamma)^2\right]$$

Damped-Trend Linear Exponential Smoothing

The model equation for damped-trend linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + \phi T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)\phi T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \phi T_{t-1} + \alpha e_t$$

$$T_t = \phi T_{t-1} + \alpha\gamma e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t + \sum_{i=1}^{k} \phi^i T_t$$

The ARIMA model equivalency to damped-trend linear exponential smoothing is the ARIMA(1,1,2) model

$$(1-\phi B)(1-B)Y_t = (1-\theta_1 B - \theta_2 B^2)\epsilon_t$$

$$\theta_1 = 1 + \phi - \alpha - \alpha\gamma\phi$$

$$\theta_2 = (\alpha - 1)\phi$$

The moving-average form of the equation (assuming $|\phi| < 1$) is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))\epsilon_{t-j}$$

For damped-trend linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

$$\{0 < \phi\gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))^2\right]$$

Seasonal Exponential Smoothing

The model equation for seasonal exponential smoothing is

$$Y_t = \mu_t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1 - \alpha)L_{t-1}$$

$$S_t = \delta(Y_t - L_t) + (1 - \delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \alpha e_t$$

$$S_t = S_{t-p} + \delta(1 - \alpha)e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t + S_{t-p+k}$$

The ARIMA model equivalency to seasonal exponential smoothing is the ARIMA(0,1,p+1)(0,1,0)$_p$ model

$$(1 - B)(1 - B^p)Y_t = (1 - \theta_1 B - \theta_2 B^p - \theta_3 B^{p+1})\epsilon_t$$

$$\theta_1 = 1 - \alpha$$

$$\theta_2 = 1 - \delta(1 - \alpha)$$

$$\theta_3 = (1 - \alpha)(\delta - 1)$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha & \text{for } j \bmod p \neq 0 \\ \alpha + \delta(1-\alpha) & \text{for } j \bmod p = 0 \end{cases}$$

For seasonal exponential smoothing, the additive-invertible region is

$$\{\max(-p\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

Winters Method – Additive Version

The model equation for the additive version of Winters method is

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

$$S_t = \delta(Y_t - L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha\gamma e_t$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t + S_{t-p+k}$$

The ARIMA model equivalency to the additive version of Winters method is the ARIMA(0,1,p+1)(0,1,0)$_p$ model

$$(1-B)(1-B^p)Y_t = \left[1 - \sum_{i=1}^{p+1} \theta_i B^i\right] \epsilon_t$$

$$\theta_j = \begin{cases} 1-\alpha-\alpha\gamma & j=1 \\ -\alpha\gamma & 2 \le j \le p-1 \\ 1-\alpha\gamma-\delta(1-\alpha) & j=p \\ (1-\alpha)(\delta-1) & j=p+1 \end{cases}$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha + j\alpha\gamma & \text{for } j \text{ mod } p \neq 0 \\ \alpha + j\alpha\gamma + \delta(1-\alpha), & \text{for } j \text{ mod } p = 0 \end{cases}$$

For the additive version of Winters method (see Archibald 1990), the additive-invertible region is

$$\{\max(-\mathrm{p}\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$

$$\{0 < \alpha\gamma < 2 - \alpha - \delta(1-\alpha)(1-cos(\vartheta)\}$$

where ϑ is the smallest nonnegative solution to the equations listed in Archibald (1990).

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

Winters Method – Multiplicative Version

In order to use the multiplicative version of Winters method, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of Winters method is

$$Y_t = (\mu_t + \beta_t t)s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t/S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

$$S_t = \delta(Y_t/L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t/S_{t-p}$$

$$T_t = T_{t-1} + \alpha\gamma e_t/S_{t-p}$$

$$S_t = S_{t-p} + \delta(1-\alpha)e_t/L_t$$

(Note: For missing values, $e_t = 0$.)

The *k*-step prediction equation is

$$\hat{Y}_t(k) = (L_t + kT_t)S_{t-p+k}$$

The multiplicative version of Winters method does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of Winters method described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[\sum_{i=0}^{\infty}\sum_{j=0}^{p-1}(\psi_{j+ip}S_{t+k}/S_{t+k-j})^2\right]$$

where ψ_j are as described for the additive version of Winters method, and $\psi_j = 0$ for $j \geq k$.

ARIMA Models

Auto**R**egressive **I**ntegrated **M**oving-**A**verage, or *ARIMA*, models predict values of a dependent time series with a linear combination of its own past values, past errors (also called shocks or innovations), and current and past values of other time series (predictor time series).

The Time Series Forecasting System uses the ARIMA procedure of SAS/ETS software to fit and forecast ARIMA models. The maximum likelihood method is used for parameter estimation. Refer to Chapter 11, "The ARIMA Procedure," for details of ARIMA model estimation and forecasting.

This section summarizes the notation used for ARIMA models.

Notation for ARIMA Models

A dependent time series that is modeled as a linear combination of its own past values and past values of an error series is known as a (pure) ARIMA model.

Nonseasonal ARIMA Model Notation

The order of an ARIMA model is usually denoted by the notation ARIMA(*p*,*d*,*q*), where

p	is the order of the autoregressive part.
d	is the order of the differencing (rarely should $d > 2$ be needed).
q	is the order of the moving-average process.

Given a dependent time series $\{Y_t : 1 \leq t \leq n\}$, mathematically the ARIMA model is written as

$$(1-B)^d Y_t = \mu + \frac{\theta(B)}{\phi(B)} a_t$$

where

t	indexes time
μ	is the mean term
B	is the backshift operator; that is, $BX_t = X_{t-1}$
$\phi(B)$	is the autoregressive operator, represented as a polynomial in the back shift operator: $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$
$\theta(B)$	is the moving-average operator, represented as a polynomial in the back shift operator: $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$
a_t	is the independent disturbance, also called the random error.

For example, the mathematical form of the ARIMA(1,1,2) model is

$$(1-B)Y_t = \mu + \frac{(1 - \theta_1 B - \theta_2 B^2)}{(1 - \phi_1 B)} a_t$$

Seasonal ARIMA Model Notation

Seasonal ARIMA models are expressed in factored form by the notation ARIMA(p,d,q)(P,D,Q)$_s$, where

P	is the order of the seasonal autoregressive part
D	is the order of the seasonal differencing (rarely should D > 1 be needed)
Q	is the order of the seasonal moving-average process
s	is the length of the seasonal cycle.

Given a dependent time series $\{Y_t : 1 \leq t \leq n\}$, mathematically the ARIMA seasonal model is written as

$$(1-B)^d(1-B^s)^D Y_t = \mu + \frac{\theta(B)\theta_s(B^s)}{\phi(B)\phi_s(B^s)} a_t$$

where

$\phi_s(B^s)$	is the seasonal autoregressive operator, represented as a polynomial in the back shift operator: $\phi_s(B^s) = 1 - \phi_{s,1}B^s - \ldots - \phi_{s,P}B^{sP}$
$\theta_s(B^s)$	is the seasonal moving-average operator, represented as a polynomial in the back shift operator: $\theta_s(B^s) = 1 - \theta_{s,1}B^s - \ldots - \theta_{s,Q}B^{sQ}$

For example, the mathematical form of the ARIMA(1,0,1)(1,1,2)$_{12}$ model is

$$(1-B^{12})Y_t = \mu + \frac{(1-\theta_1 B)(1-\theta_{s,1}B^{12} - \theta_{s,2}B^{24})}{(1-\phi_1 B)(1-\phi_{s,1}B^{12})} a_t$$

Abbreviated Notation for ARIMA Models

If the differencing order, autoregressive order, or moving-average order is zero, the notation is further abbreviated as

I(d)(D)$_s$	integrated model or ARIMA(0,d,0)(0,D,0)
AR(p)(P)$_s$	autoregressive model or ARIMA(p,0,0)(P,0,0)
IAR(p,d)(P,D)$_s$	integrated autoregressive model or ARIMA(p,d,0)(P,D,0)$_s$
MA(q)(Q)$_s$	moving average model or ARIMA(0,0,q)(0,0,Q)$_s$
IMA(d,q)(D,Q)$_s$	integrated moving average model or ARIMA(0,d,q)(0,D,Q)$_s$
ARMA(p,q)(P,Q)$_s$	autoregressive moving-average model or ARIMA(p,0,q)(P,0,Q)$_s$.

Notation for Transfer Functions

A transfer function can be used to filter a predictor time series to form a dynamic regression model.

Let Y_t be the dependent series, and let X_t be the predictor series, and let $\Psi(B)$ be a linear filter or transfer function for the effect of X_t on Y_t. The ARIMA model is then

$$(1-B)^d(1-B^s)^D Y_t = \mu + \Psi(B)(1-B)^d(1-B^s)^D X_t + \frac{\theta(B)\theta_s(B^s)}{\phi(B)\phi_s(B^s)}a_t$$

This model is called a *dynamic regression* of Y_t on X_t.

Nonseasonal Transfer Function Notation

Given the *i*th predictor time series $\{X_{i,t} : 1 \leq t \leq n\}$, the transfer function is written as [Dif(d_i)Lag(k_i)N(q_i)/ D(p_i)] where

d_i	is the simple order of the differencing for the *i*th predictor time series, $(1-B)^{d_i}X_{i,t}$ (rarely should d_i > 2 be needed).
k_i	is the pure time delay (lag) for the effect of the *i*th predictor time series, $X_{i,t}B^{k_i} = X_{i,t-k_i}$.
p_i	is the simple order of the denominator for the *i*th predictor time series.
q_i	is the simple order of the numerator for the *i*th predictor time series.

The mathematical notation used to describe a transfer function is

$$\Psi_i(B) = \frac{\omega_i(B)}{\delta_i(B)}(1-B)^{d_i}B^{k_i}$$

where

B	is the backshift operator; that is, $BX_t = X_{t-1}$.
$\delta_i(B)$	is the denominator polynomial of the transfer function for the *i*th predictor time series: $\delta_i(B) = 1 - \delta_{i,1}B - \ldots - \delta_{i,p_i}B^{p_i}$.
$\omega_i(B)$	is the numerator polynomial of the transfer function for the *i*th predictor time series: $\omega_i(B) = 1 - \omega_{i,1}B - \ldots - \omega_{i,q_i}B^{q_i}$.

The numerator factors for a transfer function for a predictor series are like the MA part of the ARMA model for the noise series. The denominator factors for a transfer function for a predictor series are like the AR part of the ARMA model for the noise series. Denominator factors introduce exponentially weighted, infinite distributed lags into the transfer function.

For example, the transfer function for the *i*th predictor time series with

$k_i = 3$	time lag is 3
$d_i = 1$	simple order of differencing is one
$p_i = 1$	simple order of the denominator is one
$q_i = 2$	simple order of the numerator is two

would be written as [Dif(1)Lag(3)N(2)/D(1)]. The mathematical notation for the transfer function in this example is

$$\Psi_i(B) = \frac{(1-\omega_{i,1}B-\omega_{i,2}B^2)}{(1-\delta_{i,1}B)}(1-B)B^3$$

Seasonal Transfer Function Notation

The general transfer function notation for the *i*th predictor time series $X_{i,t}$ with seasonal factors is [Dif(d_i)(D_i)$_s$ Lag(k_i) N(q_i)(Q_i)$_s$/ D(p_i)(P_i)$_s$] where

D_i	is the seasonal order of the differencing for the *i*th predictor time series (rarely should $D_i > 1$ be needed).
P_i	is the seasonal order of the denominator for the *i*th predictor time series (rarely should $P_i > 2$ be needed).
Q_i	is the seasonal order of the numerator for the *i*th predictor time series, (rarely should $Q_i > 2$ be needed).
s	is the length of the seasonal cycle.

The mathematical notation used to describe a seasonal transfer function is

$$\Psi_i(B) = \frac{\omega_i(B)\omega_{s,i}(B^s)}{\delta_i(B)\delta_{s,i}(B^s)}(1-B)^{d_i}(1-B^s)^{D_i}B^{k_i}$$

where

$\delta_{s,i}(B^s)$	is the denominator seasonal polynomial of the transfer function for the *i*th predictor time series: $\delta_{s,i}(B) = 1-\delta_{s,i,1}B - \ldots - \delta_{s,i,P_i}B^{sP_i}$
$\omega_{s,i}(B^s)$	is the numerator seasonal polynomial of the transfer function for the *i*th predictor time series: $\omega_{s,i}(B) = 1-\omega_{s,i,1}B - \ldots - \omega_{s,i,Q_i}B^{sQ_i}$

For example, the transfer function for the *i*th predictor time series $X_{i,t}$ whose seasonal cycle $s = 12$ with

$d_i = 2$	simple order of differencing is two
$D_i = 1$	seasonal order of differencing is one
$q_i = 2$	simple order of the numerator is two

$Q_i = 1$	seasonal order of the numerator is one

would be written as [Dif(2)(1)$_s$ N(2)(1)$_s$]. The mathematical notation for the transfer function in this example is

$$\Psi_i(B) = (1 - \omega_{i,1}B - \omega_{i,2}B^2)(1 - \omega_{s,i,1}B^{12})(1-B)^2(1-B^{12})$$

Note: In this case, [Dif(2)(1)$_s$ N(2)(1)$_s$] = [Dif(2)(1)$_s$Lag(0)N(2)(1)$_s$/D(0)(0)$_s$].

Predictor Series

This section discusses time trend curves, seasonal dummies, interventions, and adjustments.

Time Trend Curves

When you specify a time trend curve as a predictor in a forecasting model, the system computes a predictor series that is a deterministic function of time. This variable is then included in the model as a regressor, and the trend curve is fit to the dependent series by linear regression, in addition to other predictor series.

Some kinds of nonlinear trend curves are fit by transforming the dependent series. For example, the exponential trend curve is actually a linear time trend fit to the logarithm of the series. For these trend curve specifications, the series transformation option is set automatically, and you cannot independently control both the time trend curve and transformation option.

The computed time trend variable is included in the output data set in a variable named in accordance with the trend curve type. Let *t* represent the observation count from the start of the period of fit for the model, and let X_t represent the value of the time trend variable at observation *t* within the period of fit. The names and definitions of these variables are as follows. (Note: These deterministic variables are reserved variable names.)

Linear Trend	Variable name _LINEAR_, with $X_t = t - c$.
Quadratic Trend	Variable name _QUAD_, with $X_t = (t-c)^2$. Note that a quadratic trend implies a linear trend as a special case and results in two regressors: _QUAD_ and _LINEAR_.
Cubic Trend	Variable name _CUBE_, with $X_t = (t-c)^3$. Note that a cubic trend implies a quadratic trend as a special case and results in three regressors: _CUBE_, _QUAD_, and _LINEAR_.
Logistic Trend	Variable name _LOGIT_, with $X_t = t$. The model is a linear time trend applied to the logistic transform of the dependent series. Thus, specifying a logistic trend is equivalent to specifying the Logistic series transformation and

	a linear time trend. A logistic trend predictor can be used only in conjunction with the logistic transformation, which is set automatically when you specify logistic trend.
Logarithmic Trend	Variable name _LOG_, with $X_t = ln(t)$.
Exponential Trend	Variable name _EXP_, with $X_t = t$. The model is a linear time trend applied to the logarithms of the dependent series. Thus, specifying an exponential trend is equivalent to specifying the log series transformation and a linear time trend. An exponential trend predictor can be used only in conjunction with the log transformation, which is set automatically when you specify exponential trend.
Hyperbolic Trend	Variable name _HYP_, with $X_t = 1/t$.
Power Curve Trend	Variable name _POW_, with $X_t = ln(t)$. The model is a logarithmic time trend applied to the logarithms of the dependent series. Thus, specifying a power curve is equivalent to specifying the log series transformation and a logarithmic time trend. A power curve predictor can be used only in conjunction with the log transformation, which is set automatically when you specify a power curve trend.
EXP(A+B/TIME) Trend	Variable name _ERT_, with $X_t = 1/t$. The model is a hyperbolic time trend applied to the logarithms of the dependent series. Thus, specifying this trend curve is equivalent to specifying the log series transformation and a hyperbolic time trend. This trend curve can be used only in conjunction with the log transformation, which is set automatically when you specify this trend.

Intervention Effects

Interventions are used for modeling events that occur at specific times. That is, they are known changes that affect the dependent series or outliers.

The *i*th intervention series is included in the output data set with variable name _INTV*i*_, which is a reserved variable name.

Point Interventions

The point intervention is a one-time event. The *i*th intervention series $X_{i,t}$ has a point intervention at time t_{int} when the series is nonzero only at time t_{int}, that is,

$$X_{i,t} = \begin{cases} 1, & t = t_{int} \\ 0, & otherwise \end{cases}$$

Step Interventions

Step interventions are continuing, and the input time series flags periods after the intervention. For a step intervention, before time t_{int}, the *i*th intervention series $X_{i,t}$ is zero and then steps to a constant level thereafter, that is,

$$X_{i,t} = \begin{cases} 1, & t \geq t_{int} \\ 0, & otherwise \end{cases}$$

Ramp Interventions

A ramp intervention is a continuing intervention that increases linearly after the intervention time. For a ramp intervention, before time t_{int}, the *i*th intervention series $X_{i,t}$ is zero and increases linearly thereafter, that is, proportional to time.

$$X_{i,t} = \begin{cases} t - t_{int}, & t \geq t_{int} \\ 0, & otherwise \end{cases}$$

Intervention Effect

Given the *i*th intervention series $X_{i,t}$, you can define how the intervention takes effect by filters (transfer functions) of the form

$$\Psi_i(B) = \frac{1 - \omega_{i,1}B - \ldots - \omega_{i,q_i}B^{q_i}}{1 - \delta_{i,1}B - \ldots - \delta_{i,p_i}B^{p_i}}$$

where B is the backshift operator $By_t = y_{t-1}$.

The denominator of the transfer function determines the decay pattern of the intervention effect, whereas the numerator terms determine the size of the intervention effect time window.

For example, the following intervention effects are associated with the respective transfer functions.

Immediately	$\Psi_i(B) = 1$
Gradually	$\Psi_i(B) = 1/(1 - \delta_{i,1}B)$
1 Lag window	$\Psi_i(B) = 1 - \omega_{i,1}B$
3 Lag window	$\Psi_i(B) = 1 - \omega_{i,1}B - \omega_{i,2}B^2 - \omega_{i,3}B^3$

Intervention Notation

The notation used to describe intervention effects has the form *type*:t_{int}(q_i)/(p_i), where *type* is point, step, or ramp; t_{int} is the time of the intervention (for example, OCT87); q_i is the transfer function numerator order; and p_i is the transfer function denominator order. If $q_i = 0$, the part "(q_i)" is omitted; if $p_i = 0$, the part "/(p_i)" is omitted.

In the Intervention Specification window, the `Number of Lags` option specifies the transfer function numerator order q_i, and the `Effect Decay Pattern` option specifies the transfer function denominator order p_i. In the `Effect Decay Pattern options`, values and resulting p_i are: `None`, $p_i = 0$; `Exp`, $p_i = 1$; `Wave`, $p_i = 2$.

For example, a step intervention with date 08MAR90 and effect pattern `Exp` is denoted "Step:08MAR90/(1)" and has a transfer function filter $\Psi_i(B) = 1/(1 - \delta_1 B)$. A ramp intervention immediately applied on 08MAR90 is denoted "Ramp:08MAR90" and has a transfer function filter $\Psi_i(B) = 1$.

Seasonal Dummy Inputs

For a seasonal cycle of length *s*, the seasonal dummy regressors include $\{X_{i,t} : 1 \leq i \leq (s-1), 1 \leq t \leq n\}$ for models that include an intercept term and $\{X_{i,t} : 1 \leq i \leq s, 1 \leq t \leq n\}$ for models that exclude an intercept term. Each element of a seasonal dummy regressor is either zero or one, based on the following rule:

$$X_{i,t} = \begin{cases} 1, & \text{when } i = t \bmod s \\ 0, & otherwise \end{cases}$$

Note that if the model includes an intercept term, the number of seasonal dummy regressors is one less than *s* to ensure that the linear system is full rank.

The seasonal dummy variables are included in the output data set with variable names prefixed with "SDUMMY*i*" and sequentially numbered. They are reserved variable names.

Series Diagnostic Tests

This section describes the diagnostic tests that are used to determine the kinds of forecasting models appropriate for a series.

The series diagnostics are a set of heuristics that provide recommendations on whether or not the forecasting model should contain a log transform, trend terms, and seasonal terms. These recommendations are used by the automatic model selection process to restrict the model search to a subset of the model selection list. (You can disable this behavior using the Automatic Model Selection Options window.)

The tests that are used by the series diagnostics will not always produce the correct classification of the series. They are intended to accelerate the process of searching for a good forecasting model for the series, but you should not rely on them if finding the very best model is important to you.

If you have information about the appropriate kinds of forecasting models (perhaps from studying the plots and autocorrelations shown in the Series Viewer window), you can set the series diagnostic flags in the Series Diagnostics window. Select the YES, NO, or MAYBE values for the `Log Transform`, `Trend`, and `Seasonality` options in the Series Diagnostics window as you think appropriate.

The series diagnostics tests are intended as a heuristic tool only, and no statistical validity is claimed for them. These tests may be modified and enhanced in future releases of the Time Series Forecasting System. The testing strategy is as follows:

1. **Log transform test.** The log test fits a high order autoregressive model to the series and to the log of the series and compares goodness-of-fit measures for the prediction errors of the two models. If this test finds that log transforming the series is suitable, the `Log Transform` option is set to YES, and the subsequent diagnostic tests are performed on the log transformed series.
2. **Trend test.** The resultant series is tested for presence of a trend using an augmented Dickey-Fuller test and a random walk with drift test. If either test finds that the series appears to have a trend, the `Trend` option is set to YES, and the subsequent diagnostic tests are performed on the differenced series.
3. **Seasonality test.** The resultant series is tested for seasonality. A seasonal dummy model with AR(1) errors is fit and the joint significance of the seasonal dummy estimates is tested. If the seasonal dummies are significant, the AIC statistic for this model is compared to the AIC for and AR(1) model without seasonal dummies. If the AIC for the seasonal model is lower than that of the nonseasonal model, the `Seasonal` option is set to YES.

Statistics of Fit

This section explains the goodness-of-fit statistics reported to measure how well different models fit the data. The statistics of fit for the various forecasting models can be viewed or stored in a data set using the Model Viewer window.

Statistics of fit are computed using the actual and forecasted values for observations in the period of evaluation. One-step forecasted values are used whenever possible, including the case when a hold-out sample contains no missing values. If a one-step forecast for an observation cannot be computed due to missing values for previous series observations, a multi-step forecast is computed, using the minimum number of steps as the previous nonmissing values in the data range permit.

The various statistics of fit reported are as follows. In these formula, n is the number of nonmissing observations and k is the number of fitted parameters in the model.

Number of Nonmissing Observations.
The number of nonmissing observations used to fit the model.

Number of Observations.
The total number of observations used to fit the model, including both missing and nonmissing observations.

Number of Missing Actuals.
The number of missing actual values.

Number of Missing Predicted Values.
The number of missing predicted values.

Number of Model Parameters.
The number of parameters fit to the data. For combined forecast, this is the number of forecast components.

Total Sum of Squares (Uncorrected).
The total sum of squares for the series, SST, uncorrected for the mean: $\sum_{t=1}^{n} y_t^2$.

Total Sum of Squares (Corrected).
The total sum of squares for the series, SST, corrected for the mean: $\sum_{t=1}^{n} (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean.

Sum of Square Errors.
The sum of the squared prediction errors, SSE. $SSE = \sum_{t=1}^{n} (y_t - \hat{y}_t)^2$, where $\hat{y}$ is the one-step predicted value.

Mean Square Error.
The mean squared prediction error, MSE, calculated from the one-step-ahead forecasts. $MSE = \frac{1}{n}SSE$. This formula enables you to evaluate small holdout samples.

Root Mean Square Error.
The root mean square error (RMSE), $\sqrt{MSE}$.

Mean Absolute Percent Error.
The mean absolute percent prediction error (MAPE), $\frac{100}{n} \sum_{t=1}^{n} |(y_t - \hat{y}_t)/y_t|$.
The summation ignores observations where $y_t = 0$.

Mean Absolute Error.
The mean absolute prediction error, $\frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|$.

R-Square.
The R^2 statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares, *SSE*, may be larger than *SST* and the R^2 statistic will be negative.

Adjusted R-Square.
The adjusted R^2 statistic, $1 - (\frac{n-1}{n-k})(1 - R^2)$.

Amemiya's Adjusted R-Square.
Amemiya's adjusted R^2, $1 - (\frac{n+k}{n-k})(1 - R^2)$.

Random Walk R-Square.
The random walk R^2 statistic (Harvey's R^2 statistic using the random walk model for comparison), $1 - (\frac{n-1}{n})SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$, and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$.

Akaike's Information Criterion.
Akaike's information criterion (AIC), $n \ln(MSE) + 2k$.

Schwarz Bayesian Information Criterion.
Schwarz Bayesian information criterion (SBC or BIC),
$n \ln(MSE) + k \ln(n)$.

Amemiya's Prediction Criterion.
Amemiya's prediction criterion, $\frac{1}{n}SST(\frac{n+k}{n-k})(1 - R^2) = (\frac{n+k}{n-k})\frac{1}{n}SSE$.

Maximum Error.
The largest prediction error.

Minimum Error.
The smallest prediction error.

Maximum Percent Error.
The largest percent prediction error, $100\max((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

Minimum Percent Error.
The smallest percent prediction error, $100\min((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

Mean Error.
The mean prediction error, $\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)$.

Mean Percent Error.
The mean percent prediction error, $\frac{100}{n}\sum_{t=1}^{n}\frac{(y_t - \hat{y}_t)}{y_t}$. The summation ignores observations where $y_t = 0$.

References

Akaike, H. (1974), "A New Look at the Statistical Model Identification," *IEEE Transaction on Automatic Control*, AC-19, 716-723.

Aldrin, M. and Damsleth, E. (1989), "Forecasting Non-seasonal Time Series with Missing Observations," *Journal of Forecasting*, 8, 97-116.

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, New York: John Wiley & Sons, Inc.

Ansley, C. (1979), "An Algorithm for the Exact Likelihood of a Mixed Autoregressive-Moving Average Process," *Biometrika*, 66, 59.

Ansley, C. and Newbold, P. (1980), "Finite Sample Properties of Estimators for Autoregressive Moving Average Models," *Journal of Econometrics*, 13, 159.

Archibald, B.C. (1990), "Parameter Space of the Holt-Winters' Model," *International Journal of Forecasting*, 6, 199-209.

Bartolomei, S.M. and Sweet, A.L. (1989), "A Note on the Comparison of Exponential Smoothing Methods for Forecasting Seasonal Series," *International Journal of Forecasting*, 5, 111-116.

Bhansali, R.J. (1980), "Autoregressive and Window Estimates of the Inverse Correlation Function," *Biometrika*, 67, 551-566.

Bowerman, B.L. and O'Connell, R.T. (1979), *Time Series and Forecasting: An Applied Approach*, North Scituate, Massachusetts: Duxbury Press.

Box, G.E.P. and Cox D.R. (1964), "An Analysis of Transformations," *Journal of Royal Statistical Society* B, No. 26, 211-243.

Box, G.E.P. and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control,* Revised Edition, San Francisco: Holden-Day.

Box, G.E.P. and Tiao, G.C. (1975), "Intervention Analysis with Applications to Economic and Environmental Problems," *JASA*, 70, 70-79.

Brocklebank, J.C. and Dickey, D.A. (1986), *SAS System for Forecasting Time Series, 1986 Edition*, Cary, North Carolina: SAS Institute Inc.

Brown, R.G. (1962), *Smoothing, Forecasting and Prediction of Discrete Time Series*, New York: Prentice-Hall.

Brown, R.G. and Meyer, R.F. (1961), "The Fundamental Theorem of Exponential Smoothing," *Operations Research*, 9, 673-685.

Chatfield, C. (1978), "The Holt-Winters Forecasting Procedure," *Applied Statistics*, 27, 264-279.

Chatfield, C., and Prothero, D.L. (1973), "Box-Jenkins Seasonal Forecasting: Problems in a Case Study," *Journal of the Royal Statistical Society, Series A*, 136, 295-315.

Chatfield, C. and Yar, M. (1988), "Holt-Winters Forecasting: Some Practical Issues," *The Statistician*, 37, 129-140.

Chatfield, C. and Yar, M. (1991), "Prediction intervals for multiplicative Holt-Winters," *International Journal of Forecasting*, 7, 31-37.

Cogger, K.O. (1974), "The Optimality of General-Order Exponential Smoothing," *Operations Research*, 22, 858.

Cox, D. R. (1961), "Prediction by Exponentially Weighted Moving Averages and Related Methods," *Journal of the Royal Statistical Society, Series B*, 23, 414-422.

Davidson, J. (1981), "Problems with the Estimation of Moving Average Models," *Journal of Econometrics*, 16, 295.

Dickey, D. A., and Fuller, W.A. (1979), "Distribution of the Estimators for Autoregressive Time Series With a Unit Root," *Journal of the American Statistical Association*, 74(366), 427-431.

Dickey, D. A., Hasza, D. P., and Fuller, W.A. (1984), "Testing for Unit Roots in Seasonal Time Series," *Journal of the American Statistical Association*, 79(386), 355-367.

Fair, R.C. (1986), "Evaluating the Predictive Accuracy of Models," in *Handbook of Econometrics*, Vol. 3., Griliches, Z. and Intriligator, M.D., eds., New York: North Holland.

Fildes, R. (1979), "Quantitative Forecasting – the State of the Art: Extrapolative Models," *Journal of Operational Research Society*, 30, 691-710.

Fuller, W.A. (1976), *Introduction to Statistical Time Series*, New York: John Wiley & Sons, Inc.

Gardner, E.S., Jr. (1984), "The Strange Case of the Lagging Forecasts," *Interfaces*, 14, 47-50.

Gardner, E.S., Jr. (1985), "Exponential Smoothing: the State of the Art," *Journal of Forecasting*, 4, 1-38.

Granger, C.W.J. and Newbold, P. (1977), *Forecasting Economic Time Series*, New York: Academic Press, Inc.

Greene, W.H. (1993), *Econometric Analysis*, Second Edition, New York: Macmillan Publishing Company.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton University Press: Princeton.

Harvey, A.C. (1981), *Time Series Models*, New York: John Wiley & Sons, Inc.

Harvey, A.C. (1984), "A Unified View of Statistical Forecasting Procedures," *Journal of Forecasting*, 3, 245-275.

Hopewood, W.S., McKeown, J.C. and Newbold, P. (1984), "Time Series Forecasting Models Involving Power Transformations," *Journal of Forecasting*, Vol 3, No. 1, 57-61.

Jones, Richard H. (1980), "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations," *Technometrics*, 22, 389-396.

Judge, G.G., Griffiths, W.E., Hill, R.C. and Lee, T.C. (1980), *The Theory and Practice of Econometrics*, New York: John Wiley & Sons.

Ledolter, J. and Abraham, B. (1984), "Some Comments on the Initialization of Exponential Smoothing," *Journal of Forecasting*, 3, 79-84.

Ljung, G.M. and Box, G.E.P. (1978), "On a Measure of Lack of Fit in Time Series Models," *Biometrika*, 65, 297-303.

Makridakis, S., Wheelwright, S.C., and McGee, V.E. (1983), *Forecasting: Methods and Applications*, Second Edition, New York: John Wiley & Sons.

McKenzie, Ed (1984), "General Exponential Smoothing and the Equivalent ARMA Process," *Journal of Forecasting*, 3, 333-344.

McKenzie, Ed (1986), "Error Analysis for Winters' Additive Seasonal Forecasting System," *International Journal of Forecasting*, 2, 373-382.

Montgomery, D.C. and Johnson, L.A. (1976), *Forecasting and Time Series Analysis*, New York: McGraw-Hill Book Co.

Morf, M., Sidhu, G.S., and Kailath, T. (1974), "Some New Algorithms for Recursive Estimation on Constant Linear Discrete Time Systems," *I.E.E.E. Transactions on Automatic Control*, AC-19, 315-323.

Nelson, C.R. (1973), *Applied Time Series for Managerial Forecasting*, San Francisco: Holden-Day.

Newbold, P. (1981), "Some Recent Developments in Time Series Analysis," *International Statistical Review*, 49, 53-66.

Newton, H. Joseph and Pagano, Marcello (1983), "The Finite Memory Prediction of Covariance Stationary Time Series," *SIAM Journal of Scientific and Statistical Computing*, 4, 330-339.

Pankratz, Alan (1983), *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*, New York: John Wiley & Sons, Inc.

Pankratz, Alan (1991), *Forecasting with Dynamic Regression Models*, New York: John Wiley & Sons, Inc.

Pankratz, A. and Dudley, U. (1987), "Forecast of Power-Transformed Series," *Journal of Forecasting*, Vol 6, No. 4, 239-248.

Pearlman, J.G. (1980), "An Algorithm for the Exact Likelihood of a High-Order Autoregressive-Moving Average Process," *Biometrika*, 67, 232-233.

Priestly, M.B. (1981), *Spectral Analysis and Time Series, Volume 1: Univariate Series*, New York: Academic Press, Inc.

Roberts, S.A. (1982), "A General Class of Holt-Winters Type Forecasting Models," *Management Science*, 28, 808-820.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *Annals of Statistics*, 6, 461-464.

Sweet, A.L. (1985), "Computing the Variance of the Forecast Error for the Holt-Winters Seasonal Models," *Journal of Forecasting*, 4, 235-243.

Winters, P.R. (1960), "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, 6, 324-342.

Yar, M. and Chatfield, C. (1990), "Prediction Intervals for the Holt-Winters Forecasting Procedure," *International Journal of Forecasting*, 6, 127-137.

Woodfield, T.J. (1987), "Time Series Intervention Analysis Using SAS Software," *Proceedings of the Twelfth Annual SAS Users Group International Conference*, 331-339. Cary, NC: SAS Institute Inc.

Part 4
Investment Analysis

Contents

Chapter 42
Overview

Chapter Contents

Chapter 42
Overview

About Investment Analysis

The Investment Analysis system is an interactive environment for the time-value of money of a variety of investments:

- Loans
- Savings
- Depreciations
- Bonds
- Generic cashflows

Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

These analyses can help answer a number of questions you may have about your investments:

- Which option is more profitable or less costly?
- Is it better to buy or rent?
- Are the extra fees for refinancing at a lower interest rate justified?
- What is the balance of this account after saving this amount periodically for so many years?
- How much is legally tax-deductible?
- Is this a reasonable price?

Investment Analysis can be beneficial to users in many industries for a variety of decisions:

- manufacturing: cost justification of automation or any capital investment, replacement analysis of major equipment, or economic comparison of alternative designs
- government: setting funds for services
- finance: investment analysis and portfolio management for fixed-income securities

Starting Investment Analysis

There are two ways to invoke Investment Analysis from the main SAS window. One way is to select **Solutions** → **Analysis** → **Investment Analysis** from the main SAS menu, as displayed in Figure 42.1.

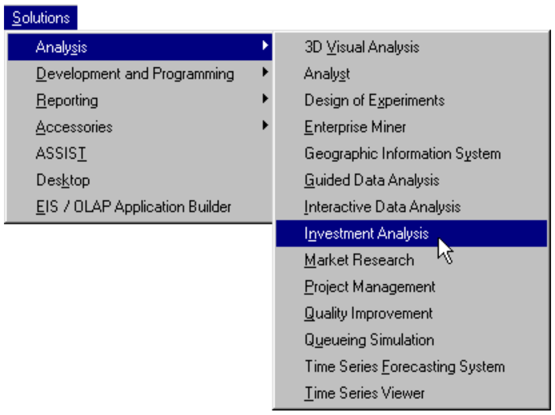


Figure 42.1. Initializing Investment Analysis with the Menu Bar

The other way is to type **INVEST** into the toolbar's command prompt, as displayed in Figure 42.2.

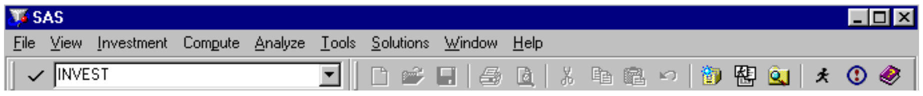

SAS
File View Investment Compute Analyze Tools Solutions Window Help
INVEST

Figure 42.2. Initializing Investment Analysis with the Toolbar

Getting Help

You can get help in Investment Analysis in three ways. One way is to use the Help Menu, as displayed in Figure 42.3. This is the right-most menu item on the menu bar.

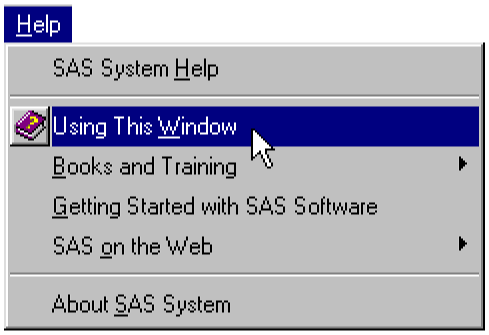


Figure 42.3. The Help Menu

Help buttons, as in Figure 42.4, provide another way to access help. Most dialog boxes provide help buttons in their lower-right corners.

Figure 42.4. A Help Button

Also, the toolbar has a button (see Figure 42.5) that invokes the help system. This is the right-most icon on the toolbar.

Figure 42.5. The Help Icon

Each of these methods invokes a browser that gives specific help for the active window.

Using Help

The chapters pertaining to Investment Analysis in this document typically have a section that introduces you to a menu and summarizes the options available through the menu. Such chapters then have sections titled Task and Dialog Box Guides. The Task section provides a description of how to perform many useful tasks. The Dialog Box Guide lists all dialog boxes pertinent to those tasks and gives a brief description of each element of each dialog box.

Software Requirements

The Investment Analysis Application is available in Version 9 of the SAS System for the following platforms:

- OS/390,
- Windows NT/2000/XP,
- OpenVMS Alpha,
- Compaq's Digital UNIX,
- HP 64,
- Solaris 64,
- AIX 64,
- RedHat Linux on Intel,
- HP/UX for Itanium Platform Family, and
- Windows for IPF.

Investment Analysis uses the following SAS software:

- Base SAS,
- SAS/ETS, and
- SAS/GRAPH (optional, to view bond pricing and breakeven graphs).

Chapter 43
Portfolios

Chapter Contents

Chapter 43
Portfolios

The File Menu

Investment Analysis stores portfolios as catalog entries. Portfolios contain a collection of investments, providing a structure to collect investments with a common purpose or goal (like a retirement or building fund portfolio). It may be advantageous also to collect investments into a common portfolio if they are competing investments you wish to perform a comparative analysis upon. Within this structure you can perform computations and analyses on a collection of investments in a portfolio, just as you would perform them on a single investment.

Investment Analysis provides many tools to aid in your manipulation of portfolios through the **File** menu, shown in Figure 43.1.

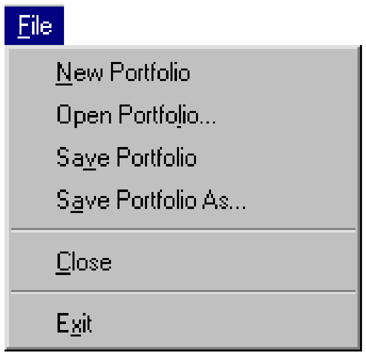


Figure 43.1. File Menu

The **File** menu offers the following items.

New Portfolio creates an empty portfolio with a new name.

Open Portfolio... opens the standard SAS Open dialog box where you select a portfolio to open.

Save Portfolio saves the current portfolio to its current name.

Save Portfolio As... opens the standard SAS Save As dialog box where you supply a new portfolio name for the current portfolio.

Close closes Investment Analysis.

Exit closes SAS (Windows only).

Tasks

Creating a New Portfolio

From the Investment Analysis dialog box, select **File** → **New Portfolio**.

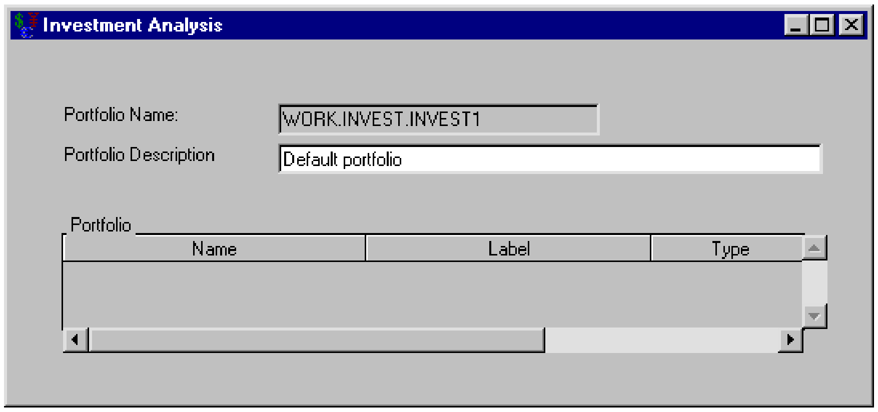


Figure 43.2. Creating a New Portfolio

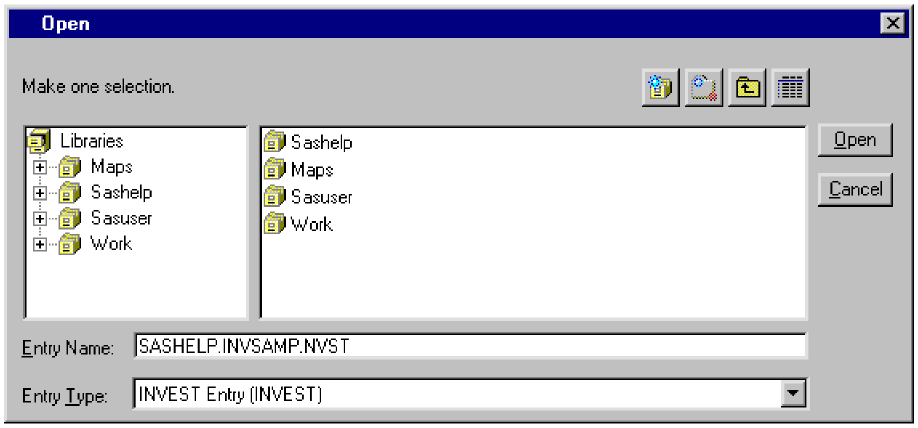
The **Portfolio Name** will be WORK.INVEST.INVEST1 as displayed in Figure 43.7, unless you have saved a portfolio to that name in the past. In that case some other unused portfolio name is given to the new portfolio.

Saving a Portfolio

From the Investment Analysis dialog box, select **File** → **Save Portfolio**. The portfolio is saved to a catalog-entry with the name in the **Portfolio Name** box.

Opening an Existing Portfolio

From the Investment Analysis dialog box, select **File** → **Open Portfolio...**. This opens the standard SAS Open dialog box. You enter the name of a SAS portfolio to open in the **Entry Name** box. For example, enter SASHELP.INVSAMP.NVST as displayed in Figure 43.3.

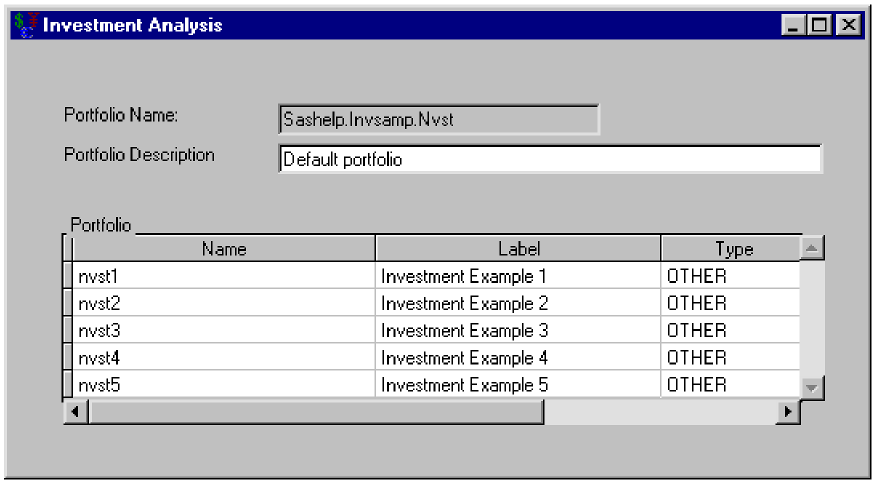


Entry Name: SASHELP.INVSAMP.NVST

Entry Type: INVEST Entry (INVEST)

Figure 43.3. Opening an Existing Portfolio

Click **Open** to load the portfolio. The portfolio should look like Figure 43.4.



Portfolio Name: Sashelp.Invsamp.Nvst

Portfolio Description: Default portfolio

Portfolio

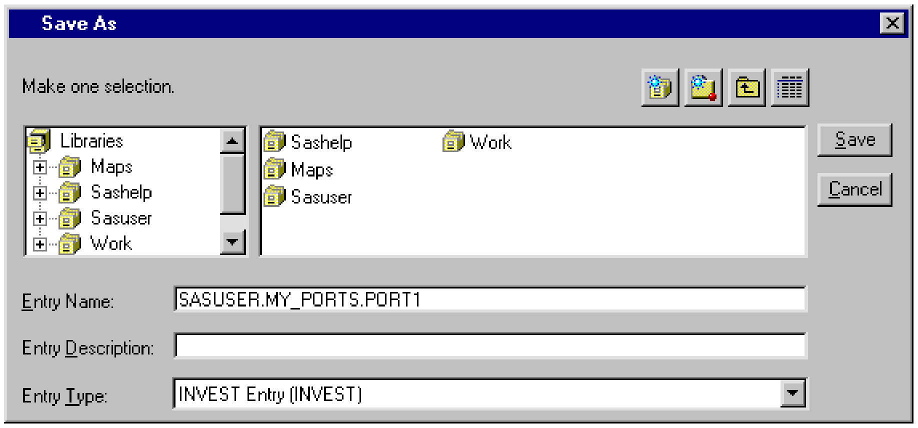
Name	Label	Type
nvst1	Investment Example 1	OTHER
nvst2	Investment Example 2	OTHER
nvst3	Investment Example 3	OTHER
nvst4	Investment Example 4	OTHER
nvst5	Investment Example 5	OTHER

Figure 43.4. The Opened Portfolio

Saving a Portfolio to a Different Name

From the Investment Analysis dialog box, select **File** → **Save Portfolio As...**.

This opens the standard SAS Save As dialog box. You can enter the name of a SAS portfolio into the **Entry Name** box. For example, enter SASUSER.MY_PORTS.PORT1 as in Figure 43.5.



Entry Name: SASUSER.MY_PORTS.PORT1

Entry Description:

Entry Type: INVEST Entry (INVEST)

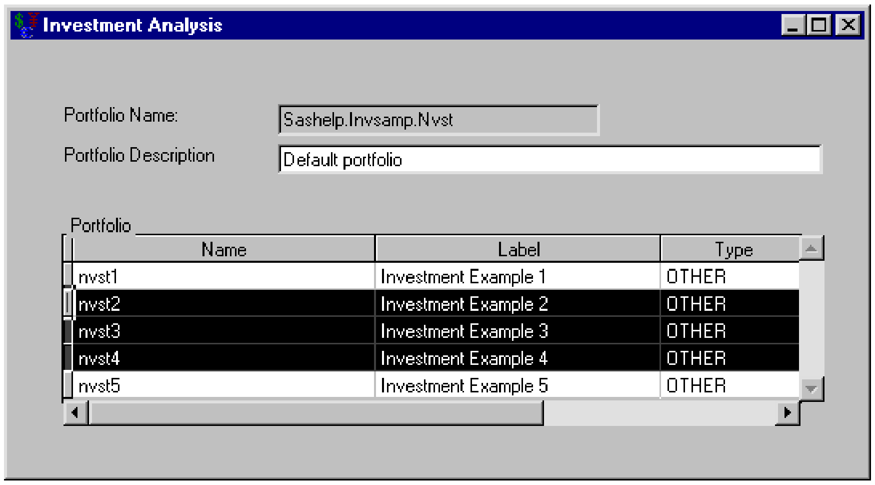
Figure 43.5. Saving a Portfolio to a Different Name

Click **Save** to save the portfolio.

Selecting Investments within a Portfolio

To select a single investment in an opened portfolio, click the investment in the Portfolio area within the Investment Analysis dialog box.

To select a list of adjacent investments, do the following: click the first investment, hold down SHIFT, and click the final investment. Once the list of investment is selected, you may release the SHIFT key. The selected investments will appear highlighted as in Figure 43.6.


Investment Analysis

Portfolio Name: Sashelp.Invsamp.Nvst

Portfolio Description: Default portfolio

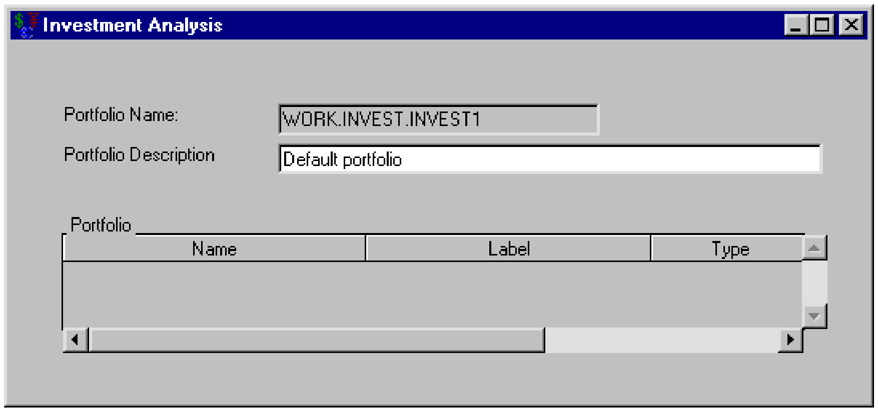
Portfolio

Name	Label	Type
nvst1	Investment Example 1	OTHER
nvst2	Investment Example 2	OTHER
nvst3	Investment Example 3	OTHER
nvst4	Investment Example 4	OTHER
nvst5	Investment Example 5	OTHER

Figure 43.6. Selecting Investments within a Portfolio

Dialog and Utility Guide

Investment Analysis


Investment Analysis

Portfolio Name: WORK.INVEST.INVEST1

Portfolio Description: Default portfolio

Portfolio

Name	Label	Type

Figure 43.7. Investment Analysis Dialog Box

Investment Portfolio Name holds the name of the portfolio. It is of the form library.catalog_entry.portfolio. The default portfolio name is work.invest.invest1, as in Figure 43.7.

Portfolio Description provides a more descriptive explanation of the portfolio's contents. You can edit this description any time this dialog box is active.

The **Portfolio** area contains the list of investments comprising the particular portfolio. Each investment in the **Portfolio** area displays the following attributes:

Name is the name of the investment. It must be a valid SAS name. It is used to distinguish investments when performing analyses and computations.

Label is a place where you can provide a more descriptive explanation of the investment.

Type is the type of investment, which is fixed when you create the investment. It will be one of the following: LOAN, SAVINGS, DEPRECIATION, BOND, or OTHER.

Additional tools to aid in the management of your portfolio are available by selecting from the menu bar and by right-clicking within the **Portfolio** area.

Menu Bar Options

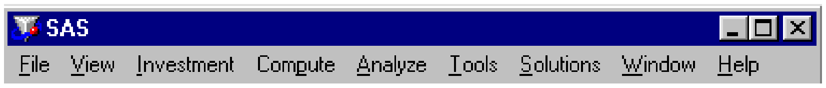

SAS
File View Investment Compute Analyze Tools Solutions Window Help

Figure 43.8. The Menu Bar

The menu bar (shown in Figure 43.8) provides many tools to aid in the management of portfolios and the investments that comprise them. The following menu items provide functionality particular to Investment Analysis:

File opens and saves portfolios.

Investment creates new investments within the portfolio.

Compute performs constant dollar, after tax, and currency conversion computations on generic cashflows.

Analyze analyzes investments to aid in decision-making.

Tools sets default values of inflation and income tax rates.

Right-Clicking within the Portfolio Area

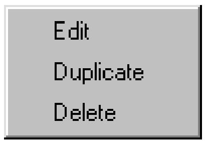


Figure 43.9. Right-Clicking

After selecting an investment, right-clicking in the **Portfolio** area pops up a menu (see Figure 43.9) that offers the following options:

Edit opens the selected investment within the portfolio.

Duplicate creates a duplicate of the selected investment within the portfolio.

Delete removes the selected investment from the portfolio.

If you wish to perform one of these actions on a collection of investments, you must select a collection of investments (as described in the section “Selecting Investments within a Portfolio” on page 2264) before right-clicking.

Chapter 44
Investments

Chapter Contents

Chapter 44
Investments

The Investment Menu

Because there are many types of investments, a tool that manages and analyzes collections of investments must be robust and flexible. Providing specifications for four specific investment types and one generic type, Investment Analysis can model almost any real-world investment.

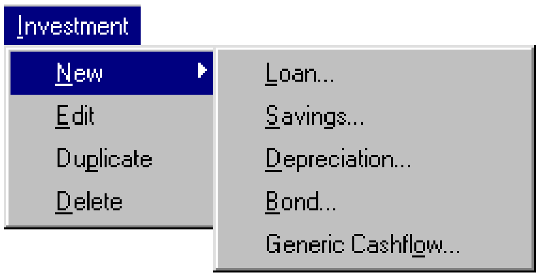


Figure 44.1. Investment Menu

The **Investment** menu, shown in Figure 44.1, offers the following items:

New → **Loan...** opens the Loan dialog box. Loans are useful for acquiring capital to pursue various interests. Available terms include rate adjustments for variable rate loans, initialization costs, prepayments, and balloon payments.

New → **Savings...** opens the Savings dialog box. Savings are necessary when planning for the future, whether for business or personal purposes. Account summary calculations available per deposit include starting balance, deposits, interest earned, and ending balance.

New → **Depreciation...** opens the Depreciation dialog box. Depreciations are relevant in tax calculation. The available depreciation methods are Straight Line, Sum-of-years Digits, Depreciation Table, and Declining Balance. Depreciation Tables are necessary when depreciation calculations must conform to set yearly percentages. Declining Balance with conversion to Straight Line is also provided.

New → **Bond...** opens the Bond dialog box. Bonds have widely varying terms depending on the issuer. As bond issuers frequently auction their bonds, the ability to price a bond between the issue date and maturity date is desirable. Fixed-coupon bonds may be analyzed for the following: price versus yield-to-maturity, duration, and convexity. These are available at different times in the bond's life.

New → **Generic Cashflow...** opens the Generic Cashflow dialog box. Generic cashflows are the most flexible investments. Only a sequence of date-amount pairs is necessary for specification. You can enter date-amount pairs and load values from

SAS data sets to specify any type of investment. You can generate uniform, arithmetic, and geometric cashflows with ease. SAS's forecasting ability is available to forecast future cashflows as well. The new graphical display aids in visualization of the cashflow and enables the user to change the frequency of the cashflow view to aggregate and disaggregate the view.

Edit opens the specification dialog box for an investment selected within the portfolio.

Duplicate creates a duplicate of an investment selected within the portfolio.

Delete removes an investment selected from the portfolio.

If you wish to edit, duplicate, or delete a collection of investments, you must select a collection of investments as described in "Selecting Investments within a Portfolio" on page 2264 before performing the menu-option.

Tasks

Loan Tasks

Suppose you want to buy a home that costs $100,000. You can make a down payment of $20,000. Hence, you need a loan of $80,000. You are able to acquire a 30-year loan at 7% interest starting January 1, 2000. Let's use Investment Analysis to specify and analyze this loan.

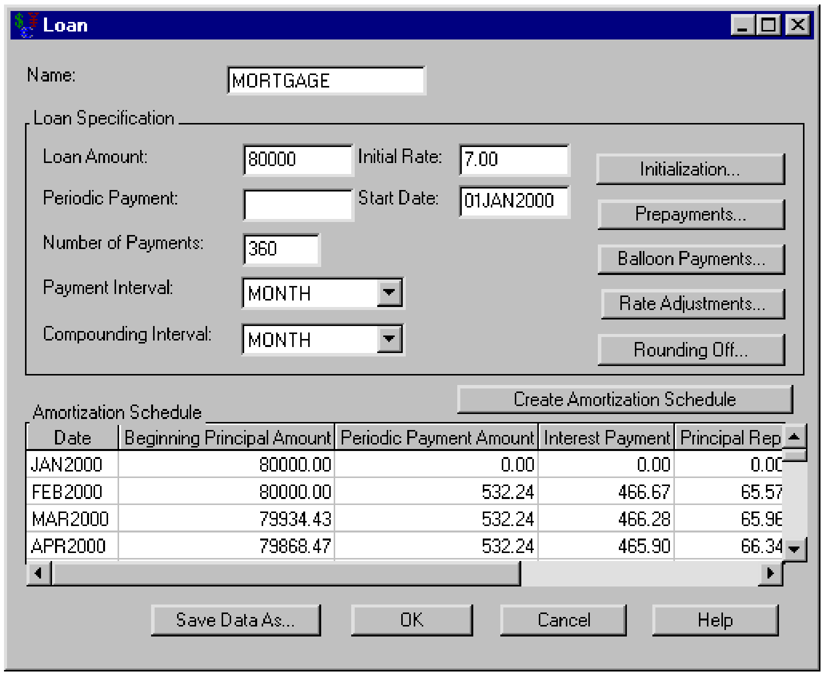
From the Investment Analysis dialog box, select **Investment** → **New** → **Loan...** from the menu bar to open the Loan dialog box.

Specifying Loan Terms to Create an Amortization Schedule

You must specify the loan before generating the amortization table. To specify the loan, follow these steps:

1. Enter **MORTGAGE** for the **Name**.
2. Enter 80000 for the **Loan Amount**.
3. Enter 7 for the **Initial Rate**.
4. Enter 360 for the **Number of Payments**.
5. Enter 01JAN2000 for the **Start Date**.

Once you have specified the loan, click **Create Amortization Schedule** to generate the amortization schedule displayed in Figure 44.2.



Loan

Name: MORTGAGE

Loan Specification

Loan Amount: 80000

Initial Rate: 7.00

Periodic Payment:

Start Date: 01JAN2000

Number of Payments: 360

Payment Interval: MONTH

Compounding Interval: MONTH

Amortization Schedule

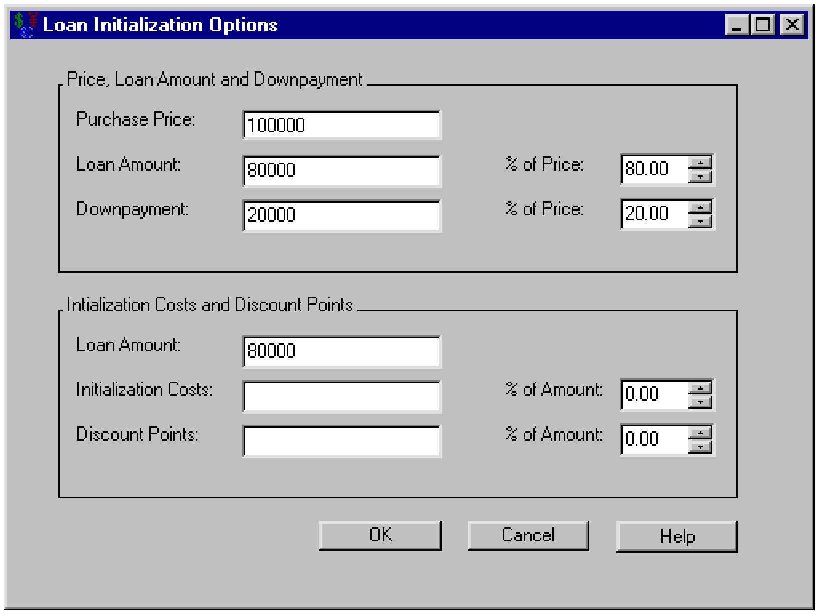
Date	Beginning Principal Amount	Periodic Payment Amount	Interest Payment	Principal Rep
JAN2000	80000.00	0.00	0.00	0.00
FEB2000	80000.00	532.24	466.67	65.57
MAR2000	79934.43	532.24	466.28	65.96
APR2000	79868.47	532.24	465.90	66.34

Figure 44.2. Creating an Amortization Schedule

Storing Other Loan Terms

Let's include information concerning the purchase price and downpayment. These terms are not necessary to specify the loan, but it may be advantageous to store such information with the loan.

Consider the loan described in "Loan Tasks" on page 2272. From the Loan dialog box (Figure 44.2) click **Initialization...** to open the Loan Initialization Options dialog box. Here you can specify the down payment, initialization costs, and discount points. To specify the down payment, enter 100000 for the **Purchase Price**, as shown in Figure 44.3.



Price, Loan Amount and Downpayment

Purchase Price:	100000		
Loan Amount:	80000	% of Price:	80.00
Downpayment:	20000	% of Price:	20.00

Intialization Costs and Discount Points

Loan Amount:	80000		
Initialization Costs:		% of Amount:	0.00
Discount Points:		% of Amount:	0.00

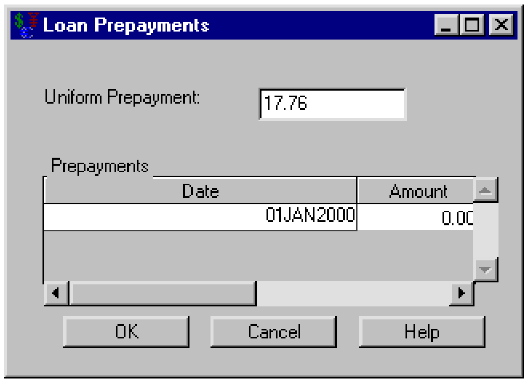
Figure 44.3. Including the Purchase Price

Click **OK** to return to the Loan dialog box.

Adding Prepayments

Now let's observe the effect of prepayments on the loan. Consider the loan described in "Loan Tasks" on page 2272. You must pay a minimum of $532.24 each month to keep up with payments. However, let's say you dislike entering this amount in your checkbook. You would rather pay $550.00 to keep the arithmetic simpler. This would constitute a uniform prepayment of $17.76 each month.

From the Loan dialog box, click **Prepayments...** that opens the Loan Prepayments dialog box shown in Figure 44.4.



Uniform Prepayment: 17.76

Prepayments

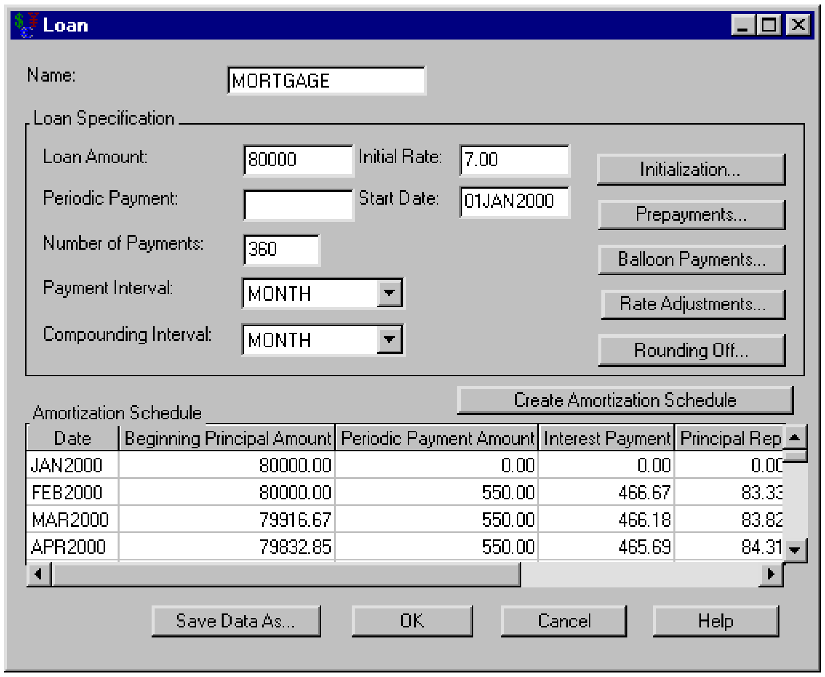
Date	Amount
01JAN2000	0.00

Figure 44.4. Specifying the Loan Prepayments

You can specify an arbitrary sequence of prepayments in the **Prepayments** area. If you want a uniform prepayment, clear the **Prepayments** area and enter the uniform payment amount in the **Uniform Prepayment** box. That amount will be added to each payment until the loan is paid off.

To specify this uniform prepayment, follow these steps:

1. Enter 17.76 for the **Uniform Prepayment**.
2. Click **OK** to return to the Loan dialog box.
3. Click **Create Amortization Schedule**, and the amortization schedule updates, as displayed in Figure 44.5.



Name: MORTGAGE

Loan Amount: 80000
Initial Rate: 7.00
Periodic Payment:
Start Date: 01JAN2000
Number of Payments: 360
Payment Interval: MONTH
Compounding Interval: MONTH

Amortization Schedule

Date	Beginning Principal Amount	Periodic Payment Amount	Interest Payment	Principal Rep
JAN2000	80000.00	0.00	0.00	0.00
FEB2000	80000.00	550.00	466.67	83.33
MAR2000	79916.67	550.00	466.18	83.82
APR2000	79832.85	550.00	465.69	84.31

Figure 44.5. The Amortization Schedule with Loan Prepayments

The last payment is on January 2030 without prepayments and February 2027 with prepayment; you would pay the loan off almost three years earlier with the $17.76 prepayments.

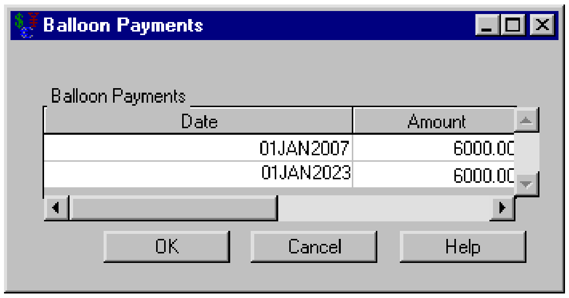
To continue this example you must remove the prepayments from the loan specification, following these steps:

1. Return to the Prepayments dialog box from the Loan dialog box by clicking **Prepayments...**.
2. Enter 0 for **Uniform Prepayment**.
3. Click **OK** to return to the Loan dialog box.

Adding Balloon Payments

Consider the loan described in "Loan Tasks" on page 2272. Suppose you cannot afford the payments of $532.24 each month. To lessen your monthly payment you could pay balloon payments of $6,000 at the end of 2007 and 2023. You wonder how this would affect your monthly payment. (Note that Investment Analysis does not allow both balloon payments and rate adjustments to be specified for a loan.)

From the Loan dialog box, click **Balloon Payments...**, which opens the Balloon Payments dialog box shown in Figure 44.6.



Balloon Payments

Date	Amount
01JAN2007	6000.0C
01JAN2023	6000.0C

Figure 44.6. Defining Loan Balloon Payments

You can specify an arbitrary sequence of balloon payments by adding date-amount pairs to the **Balloon Payments** area.

To specify these balloon payments, follow these steps:

1. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
2. Set the pair's **Date** to 01JAN2007.
3. Set its **Amount** to 6000.
4. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
5. Set the new pair's **Date** to 01JAN2023.
6. Set its **Amount** to 6000.

Click **OK** to return to the Loan dialog box. Click **Create Amortization Schedule**, and the amortization schedule updates. Your monthly payment is now $500.30, a difference of approximately $32 each month.

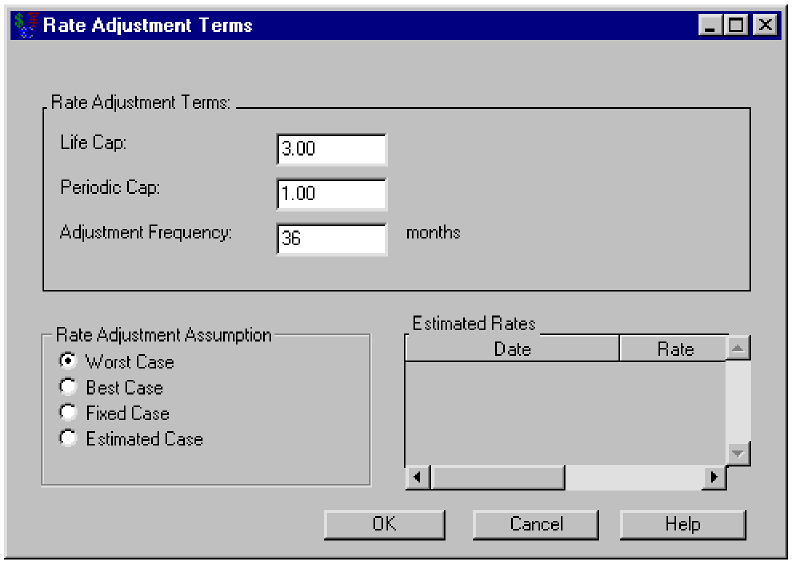
To continue this example you must remove the balloon payments from the loan specification, following these steps:

1. Return to the Balloon Payments dialog box.
2. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **Clear**.
3. Click **OK** to return to the Loan dialog box.

Handling Rate Adjustments

Consider the loan described in "Loan Tasks" on page 2272. Another option for lowering your payments is to get a variable rate loan. You can acquire a three-year-ARM at 6% with a periodic cap of 1% with a maximum of 9%. (Note that Investment Analysis does not allow both rate adjustments and balloon payments to be specified for a loan.)

From the Loan dialog box, click **Rate Adjustments...** to open the Rate Adjustment Terms dialog box shown in Figure 44.7.



Rate Adjustment Terms

Rate Adjustment Terms:

Life Cap: 3.00

Periodic Cap: 1.00

Adjustment Frequency: 36 months

Rate Adjustment Assumption

[✓ Worst Case]
Best Case
Fixed Case
Estimated Case

Estimated Rates

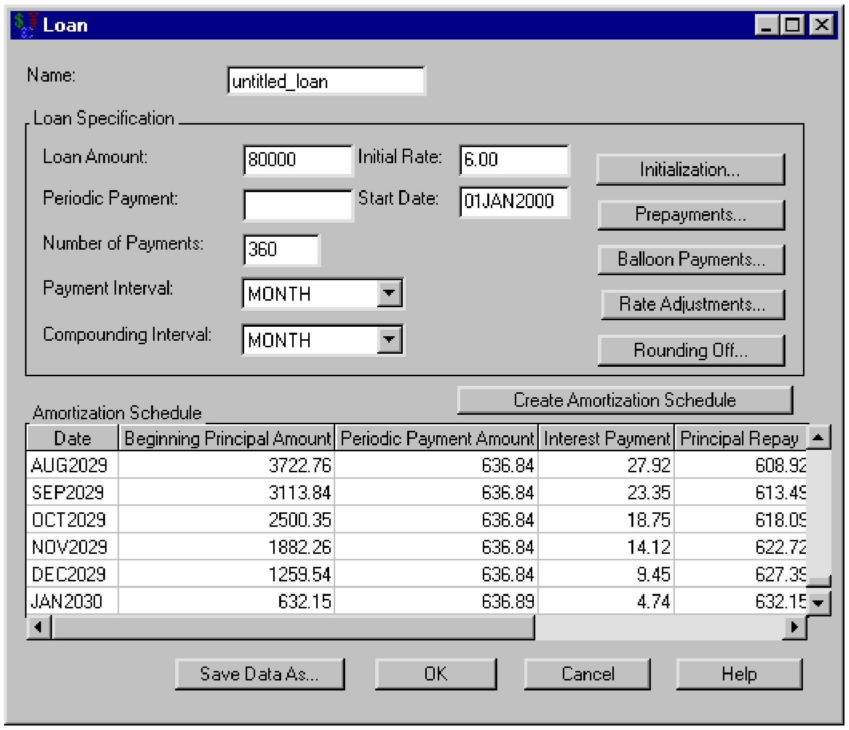
Date	Rate

Figure 44.7. Setting the Rate Adjustments

To specify these loan adjustment terms, follow these steps:

1. Enter 3 for the **Life Cap**. The **Life Cap** is the maximum deviation from the **Initial Rate**.
2. Enter 1 for the **Periodic Cap**.
3. Enter 36 for the **Adjustment Frequency**.
4. Confirm that **Worst Case** is selected in the Rate Adjustment Assumption area.
5. Click **OK** to return to the Loan dialog box.
6. Enter 6 for the **Initial Rate**.

Click **Create Amortization Schedule**, and the amortization schedule updates. Your monthly payment drops to $479.64 each month. However, if the worst-case scenario plays out, the payments will increase to $636.84 in nine years. Figure 44.8 displays amortization table information for the final few months under this scenario.



Name: untitled_loan

Loan Specification

Loan Amount:	80000	Initial Rate:	6.00
Periodic Payment:		Start Date:	01JAN2000
Number of Payments:	360		
Payment Interval:	MONTH		
Compounding Interval:	MONTH		

Amortization Schedule

Date	Beginning Principal Amount	Periodic Payment Amount	Interest Payment	Principal Repay
AUG2029	3722.76	636.84	27.92	608.92
SEP2029	3113.84	636.84	23.35	613.49
OCT2029	2500.35	636.84	18.75	618.09
NOV2029	1882.26	636.84	14.12	622.72
DEC2029	1259.54	636.84	9.45	627.39
JAN2030	632.15	636.89	4.74	632.15

Figure 44.8. The Amortization Schedule with Rate Adjustments

Click **OK** to return to the Investment Analysis dialog box.

Specifying Savings Terms to Create an Account Summary

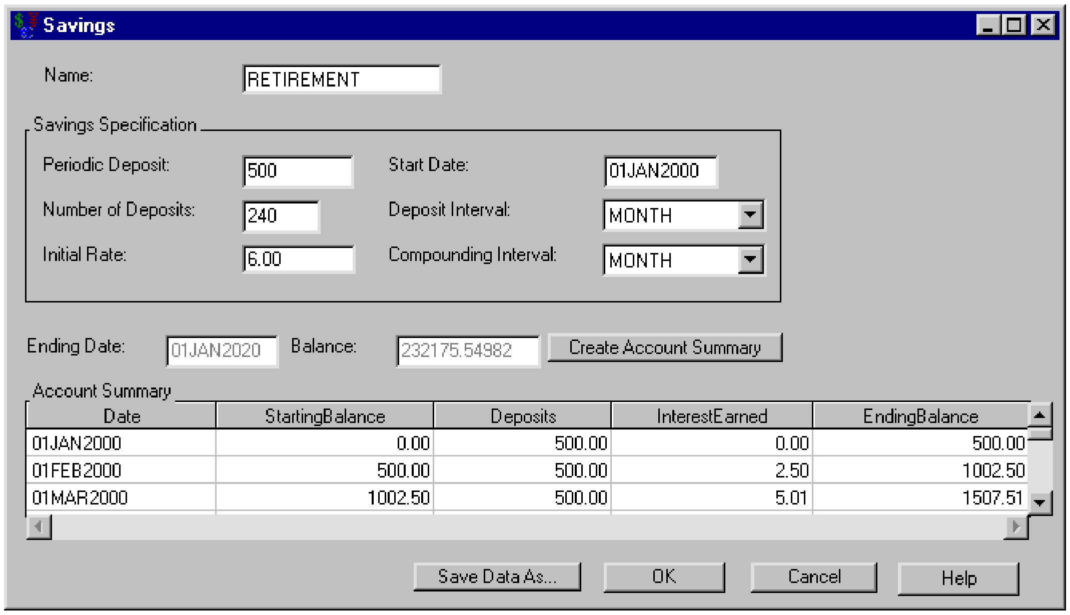
Suppose you put $500 each month into an account that earns 6% interest for 20 years. What is the balance of the account after those 20 years?

From the Investment Analysis dialog box, select **Investment** → **New** → **Savings...** from the menu bar to open the Savings dialog box.

To specify the savings, follow these steps:

1. Enter **RETIREMENT** for the **Name**.
2. Enter 500 for the **Periodic Deposit**.
3. Enter 240 for the **Number of Deposits**.
4. Enter 6 for the **Initial Rate**.

You must specify the savings before generating the account summary. Once you have specified the savings, click **Create Account Summary** to compute the ending date and balance and to generate the account summary displayed in Figure 44.9.



Name: RETIREMENT

Savings Specification

Periodic Deposit:	500	Start Date:	01JAN2000
Number of Deposits:	240	Deposit Interval:	MONTH
Initial Rate:	6.00	Compounding Interval:	MONTH

Ending Date: 01JAN2020 Balance: 232175.54982

Account Summary

Date	StartingBalance	Deposits	InterestEarned	EndingBalance
01JAN2000	0.00	500.00	0.00	500.00
01FEB2000	500.00	500.00	2.50	1002.50
01MAR2000	1002.50	500.00	5.01	1507.51

Figure 44.9. Creating an Account Summary

Click **OK** to return to the Investment Analysis dialog box.

Depreciation Tasks

Commercial assets are considered to lose value as time passes. For tax purposes, you want to quantify this loss. This investment structure helps calculate appropriate values.

Suppose you buy a boat that costs $50,000 for commercial fishing that is considered to have a ten-year useful life. How would you depreciate it?

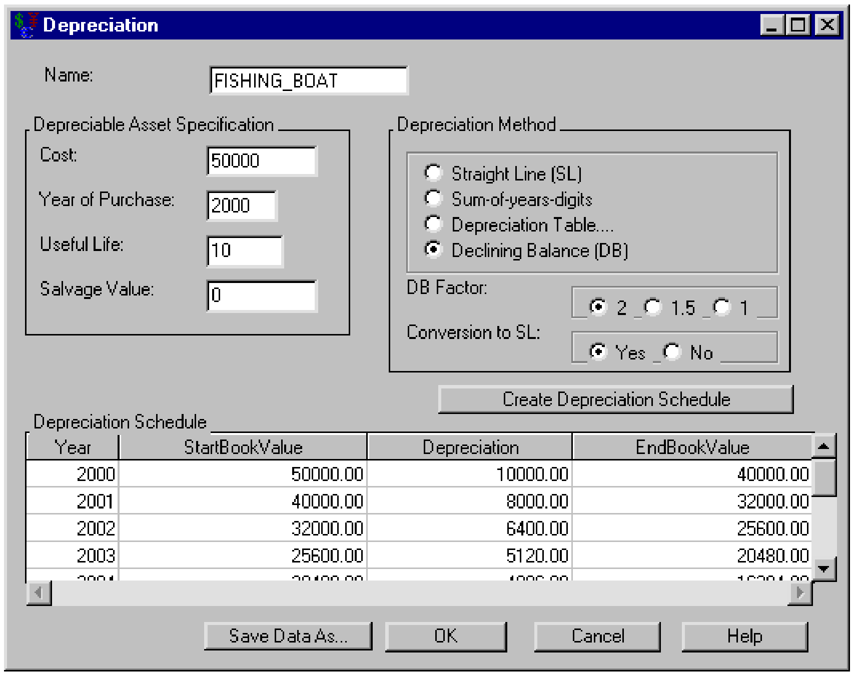
From the Investment Analysis dialog box, select **Investment** → **New** → **Depreciation...** from the menu bar to open the Depreciation dialog box.

Specifying Depreciation Terms to Create a Depreciation Table

To specify the depreciation, follow these steps:

1. Enter **FISHING_BOAT** for the **Name**.
2. Enter 50000 for the **Cost**.
3. Enter 2000 for the **Year of Purchase**.
4. Enter 10 for the **Useful Life**.
5. Enter 0 for the **Salvage Value**.

You must specify the depreciation before generating the depreciation schedule. Once you have specified the depreciation, click **Create Depreciation Schedule** to generate a deprecation schedule like the one displayed in Figure 44.10.



Name: FISHING_BOAT

Depreciable Asset Specification

Cost:	50000
Year of Purchase:	2000
Useful Life:	10
Salvage Value:	0

Depreciation Method

- Straight Line (SL)
- Sum-of-years-digits
- Depreciation Table....
- [✓ Declining Balance (DB)]

DB Factor: [✓ 2] 1.5 1

Conversion to SL: [✓ Yes] No

Depreciation Schedule

Year	StartBookValue	Depreciation	EndBookValue
2000	50000.00	10000.00	40000.00
2001	40000.00	8000.00	32000.00
2002	32000.00	6400.00	25600.00
2003	25600.00	5120.00	20480.00

Figure 44.10. Creating a Depreciation Schedule

The default deprecation method is Declining Balance (with Conversion to Straight Line). Try the following methods to see how they each affect the schedule:

- Straight Line
- Sum-of-years Digits
- Declining Balance (without conversion to Straight Line)

It might be useful to compare the value of the boat at 5 years for each method.

A description of these methods is available in "Depreciation Methods" on page 2346.

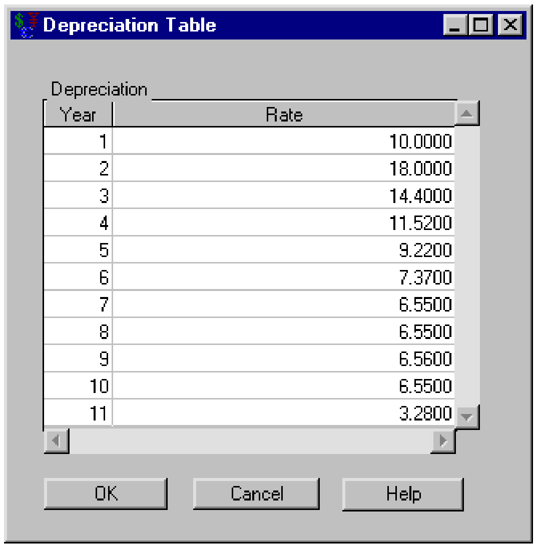
Using the Depreciation Table

Sometimes you want to force the depreciation rates to be certain percentages each year. This option is particularly useful for calculating Modified Accelerated Cost Recovery System (MACRS) Depreciations. The United States' Tax Reform Act of 1986 set depreciation rates for an asset based on an assumed lifetime for that asset. Since these lists of rates are important to many people, Investment Analysis provides SAS Datasets for situations with yearly rates (using the "half-year convention"). Find them at SASHELP.MACRS* where * refers to the class of the property. For example, use SASHELP.MACRS15 for a fifteen-year property. (When using the MACRS with the Tax Reform Act tables, you must set the **Salvage Value** to zero.)

Suppose you want to compute the depreciation schedule for the commercial fishing boat described in "Depreciation Tasks" on page 2279. The boat is a ten-year property according to the Tax Reform Act of 1986.

To employ the MACRS depreciation from the Depreciation dialog box, follow these steps:

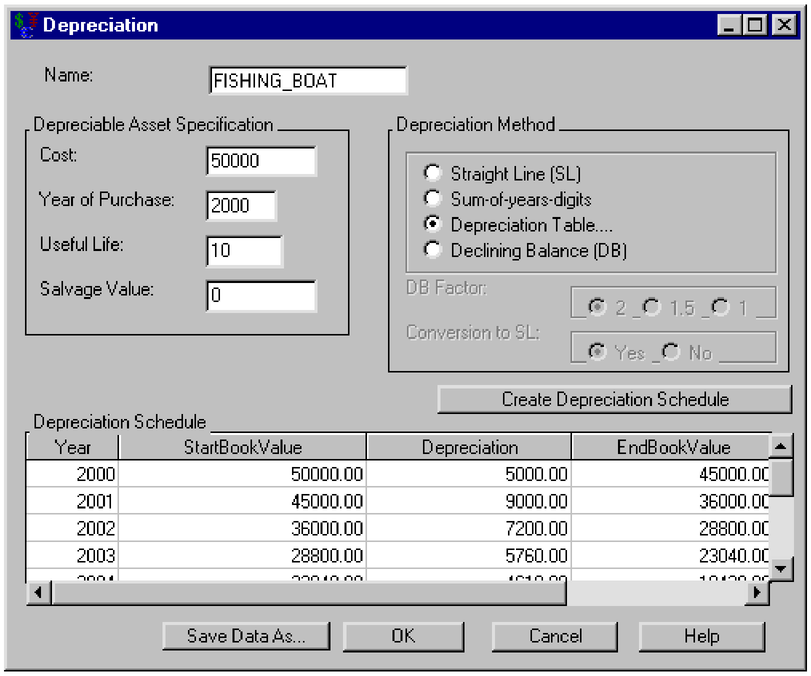
1. Click **Depreciation Table...** within the **Deprecation Method** area. This opens the Depreciation Table dialog box.
2. Right-click within the **Depreciation** area (which pops up a menu) and select **Load**.
3. Enter SASHELP.MACRS10 for the **Dataset Name**. The dialog box should look like Figure 44.11.



Year	Rate
1	10.0000
2	18.0000
3	14.4000
4	11.5200
5	9.2200
6	7.3700
7	6.5500
8	6.5500
9	6.5600
10	6.5500
11	3.2800

Figure 44.11. MACRS Percentages for a Ten-Year Property

Click **OK** to return to the Depreciation dialog box. Click **Create Depreciation Schedule** and the depreciation schedule fills (see Figure 44.12).



Name: FISHING_BOAT

Depreciable Asset Specification

Cost: 50000

Year of Purchase: 2000

Useful Life: 10

Salvage Value: 0

Depreciation Method

Straight Line (SL) Sum-of-years-digits [✓ Depreciation Table....] Declining Balance (DB)

DB Factor: [✓ 2] 1.5 1

Conversion to SL: [✓ Yes] No

Depreciation Schedule

Year	StartBookValue	Depreciation	EndBookValue
2000	50000.00	5000.00	45000.00
2001	45000.00	9000.00	36000.00
2002	36000.00	7200.00	28800.00
2003	28800.00	5760.00	23040.00

Figure 44.12. Depreciation Table with MACRS10

Note there are eleven entries in this depreciation table. This is because of the half-year convention that enables you to deduct one half of a year the first year which leaves a half year to deduct after the useful life is over.

Click **OK** to return to the Investment Analysis dialog box.

Bond Tasks

Suppose someone offers to sell you a 20-year utility bond. It was issued six years ago. It has a $1,000 face value and pays semi-year coupons at 2%. You can purchase it for $780. Would you be satisfied with this bond if you expect an 8% MARR?

From the Investment Analysis dialog box, select **Investment** → **New** → **Bond...** from the menu bar to open the Bond dialog box.

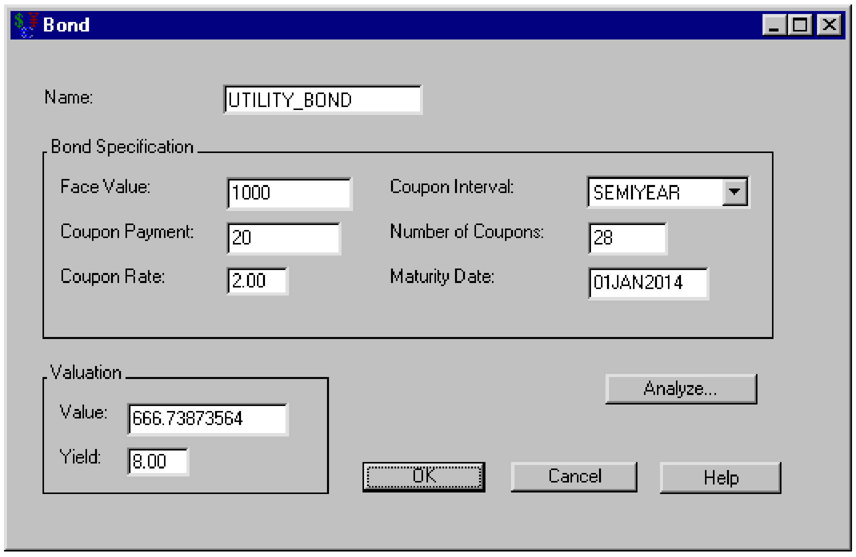
Specifying Bond Terms

To specify the bond, follow these steps:

1. Enter **UTILITY_BOND** for the **Name**.
2. Enter 1000 for the **Face Value**.
3. Enter 2 for the **Coupon Rate**. The **Coupon Payment** updates to 20.
4. Select SEMIYEAR for **Coupon Interval**.
5. Enter 28 for the **Number of Coupons**. As 14 years remain before the bond matures, the bond still has 28 semiyear coupons to pay. The **Maturity Date** updates.

Computing the Price from Yield

Enter 8 for **Yield** within the **Valuation** area. You see the bond's value would be $666.72 as in Figure 44.13.



Name: UTILITY_BOND

Bond Specification

Face Value:	1000	Coupon Interval:	SEMIYEAR
Coupon Payment:	20	Number of Coupons:	28
Coupon Rate:	2.00	Maturity Date:	01JAN2014

Valuation

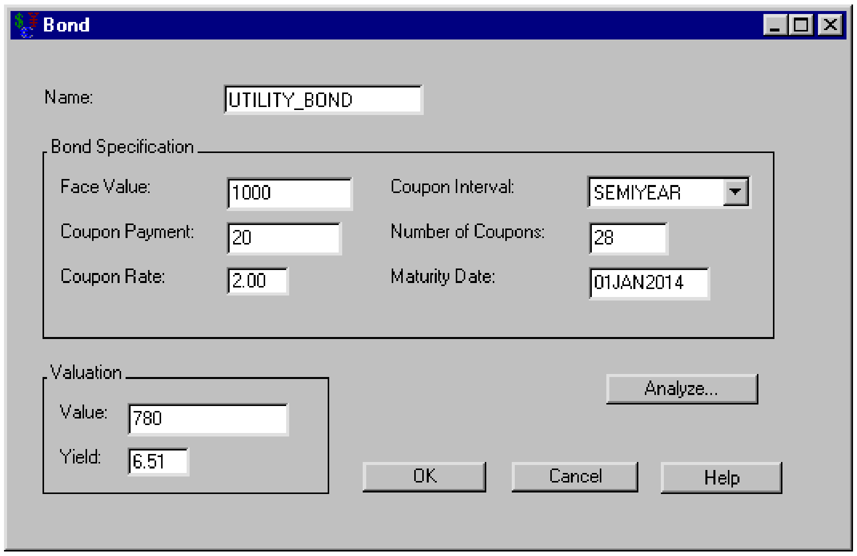
Value: 666.73873564

Yield: 8.00

Figure 44.13. Bond Value

Computing the Yield from Price

Now enter 780 for **Value** within the **Valuation** area. You see the yield is only 6.5%, as in Figure 44.14. This is not acceptable if you desire an 8% MARR.



Name: UTILITY_BOND

Bond Specification

Face Value:	1000	Coupon Interval:	SEMIYEAR
Coupon Payment:	20	Number of Coupons:	28
Coupon Rate:	2.00	Maturity Date:	01JAN2014

Valuation

Value: 780

Yield: 6.51

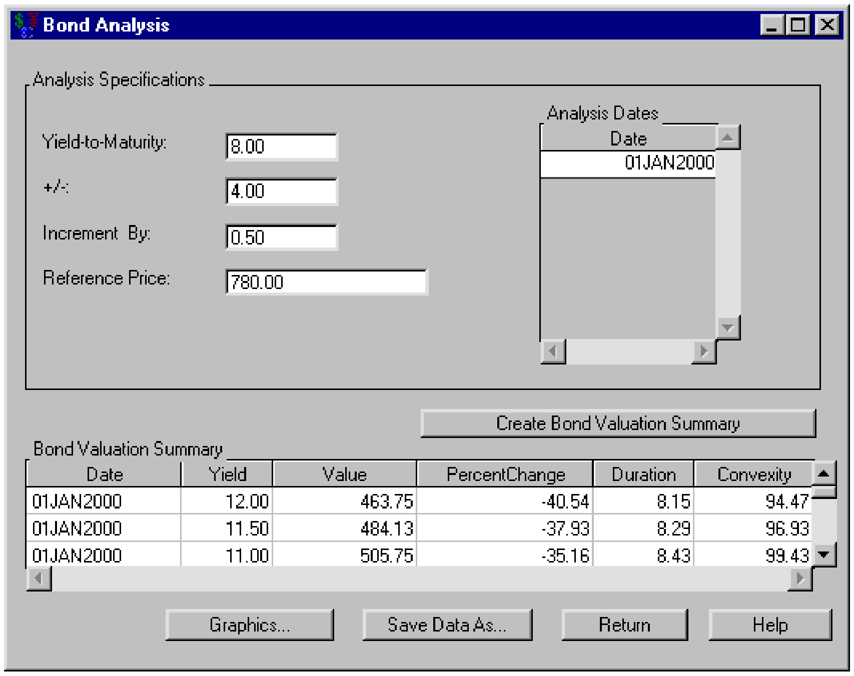
Figure 44.14. Bond Yield

Performing Bond Analysis

To perform bond-pricing analysis, follow these steps:

1. Click **Analyze...** to open the Bond Analysis dialog box.
2. Enter 8.0 as the **Yield to Maturity**.
3. Enter 4.0 as the +/-.
4. Enter 0.5 as the **Increment by**.
5. Enter 780 as the **Reference Price**.
6. Click **Create Bond Valuation Summary**.

The **Bond Valuation Summary** area fills and shows you the different values for various yields as in Figure 44.15.



Yield-to-Maturity: 8.00
+/-: 4.00
Increment By: 0.50
Reference Price: 780.00

Date
01JAN2000

Date	Yield	Value	PercentChange	Duration	Convexity
01JAN2000	12.00	463.75	-40.54	8.15	94.47
01JAN2000	11.50	484.13	-37.93	8.29	96.93
01JAN2000	11.00	505.75	-35.16	8.43	99.43

Figure 44.15. Bond Price Analysis

Creating a Price versus Yield-to-Maturity Graph

Click **Graphics...** to open the Bond Price dialog box. This contains the price versus yield-to-maturity graph shown in Figure 44.16.

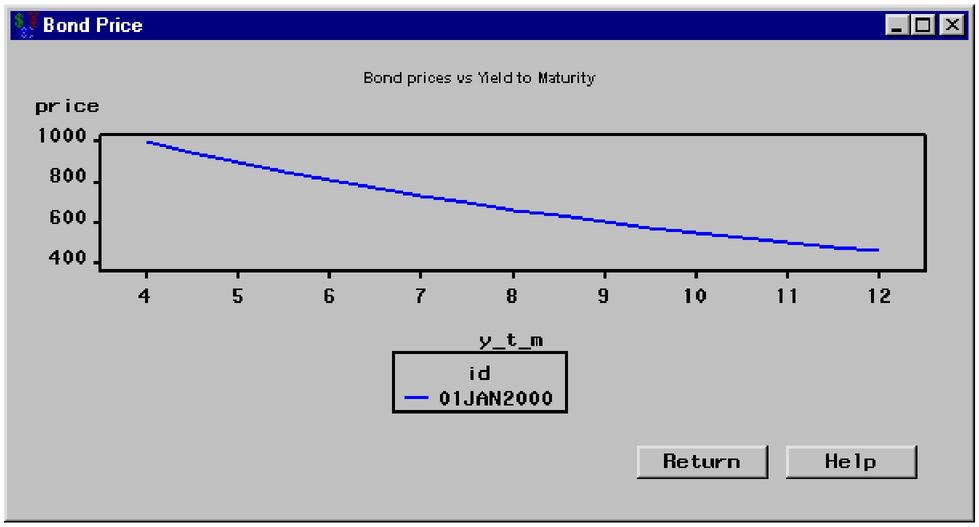


Figure 44.16. Bond Price Graph

Click **Return** to return to the Bond Analysis dialog box. Click **OK** to return to the Bond dialog box. Click **OK** to return to the Investment Analysis dialog box.

Generic Cashflow Tasks

To specify a generic cashflow, you merely define any sequence of date-amount pairs. The flexibility of generic cashflows enables the user to represent economic alternatives/investments that do not fit into loan, savings, depreciation, or bond specifications.

From the Investment Analysis dialog box, select **Investment** → **New** → **Generic Cashflow...** from the menu bar to open the **Generic Cashflow** dialog box. Enter RETAIL for the **Name** as in Figure 44.17.

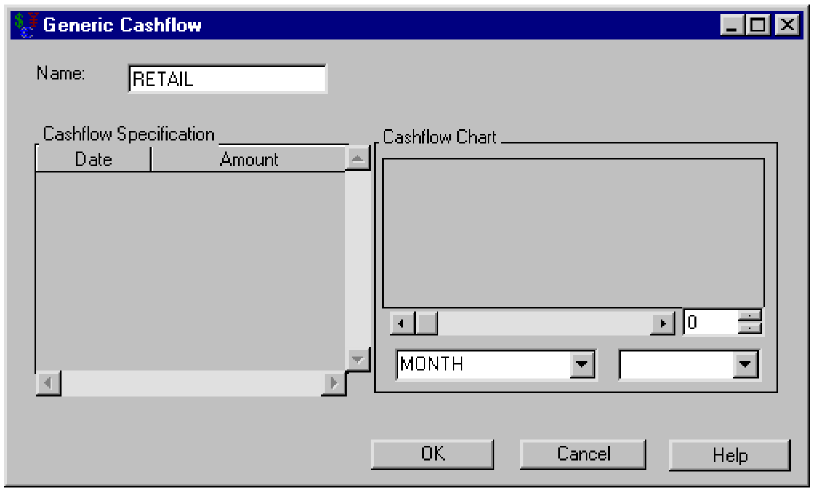

Name: RETAIL

Figure 44.17. Introducing the Generic Cashflow

Right-Clicking within the Cashflow Specification Area

Right-clicking within Generic Cashflow's **Cashflow Specification** area reveals the pop-up menu displayed in Figure 44.18. The menu provides many useful tools to assist you in creating these date-amount pairs.

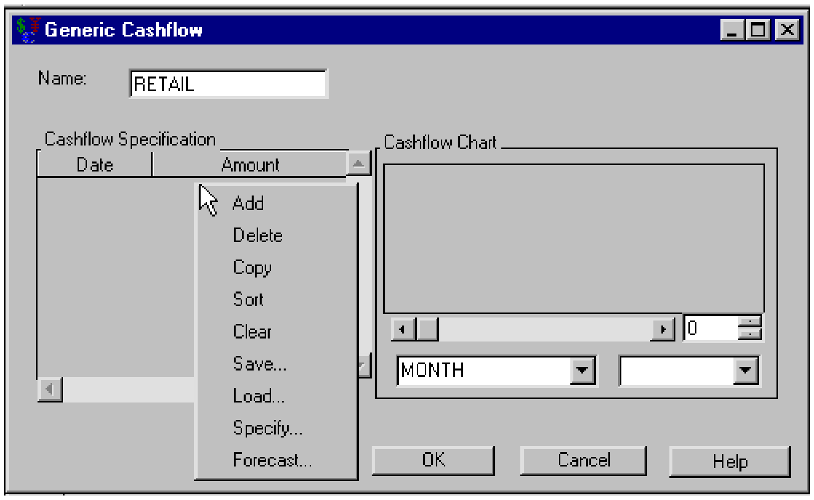


Figure 44.18. Right-Clicking within the Cashflow Specification Area

The following sections describe how to use most of these right-click options. The **Specify...** and **Forecast...** options are described in "Including a Generated Cashflow" and "Including a Forecasted Cashflow".

Adding a New Date-Amount Pair

To add a new date-amount pair manually, follow these steps:

1. Right-click in the **Cashflow Specification** area as shown in Figure 44.18, and release on **Add**.
2. Enter 01JAN01 for the date.
3. Enter 100 for the amount.

Copying a Date-Amount Pair

To copy a selected date-amount pair, follow these steps:

1. Select the pair you just created.
2. Right-click in the **Cashflow Specification** area as shown in Figure 44.18, but this time release on **Copy**.

Sorting All of the Date-Amount Pairs

Change the second date to 01JAN00. Now the dates are unsorted. Right-click in the **Cashflow Specification** area as shown in Figure 44.18, and release on **Sort**.

Deleting a Date-Amount Pair

To delete a selected date-amount pair, follow these steps:

1. Select a date-amount pair.
2. Right-click in the **Cashflow Specification** area as shown in Figure 44.18, and release on **Delete**.

Clearing All of the Date-Amount Pairs

To clear all date-amount pairs, right-click in the **Cashflow Specification** area as shown in Figure 44.18, and release on **Clear**.

Loading Date-Amount Pairs from a Dataset

To load date-amount pairs from a SAS data set into the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Load...**. This opens the Load Dataset dialog box.
2. Enter SASHELP.RETAIL for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

If there is a **Date** variable in the SAS data set, Investment Analysis loads it into the list. If there is no **Date** variable, it loads the first available date or datetime-formatted variable. Investment Analysis then searches the SAS data set for an **Amount** variable to use. If none exists, it takes the first numeric variable that is not used by the **Date** variable.

Saving Date-Amount Pairs to a Dataset

To save date-amount pairs from the **Cashflow Specification** area to a SAS data set, follow these steps:

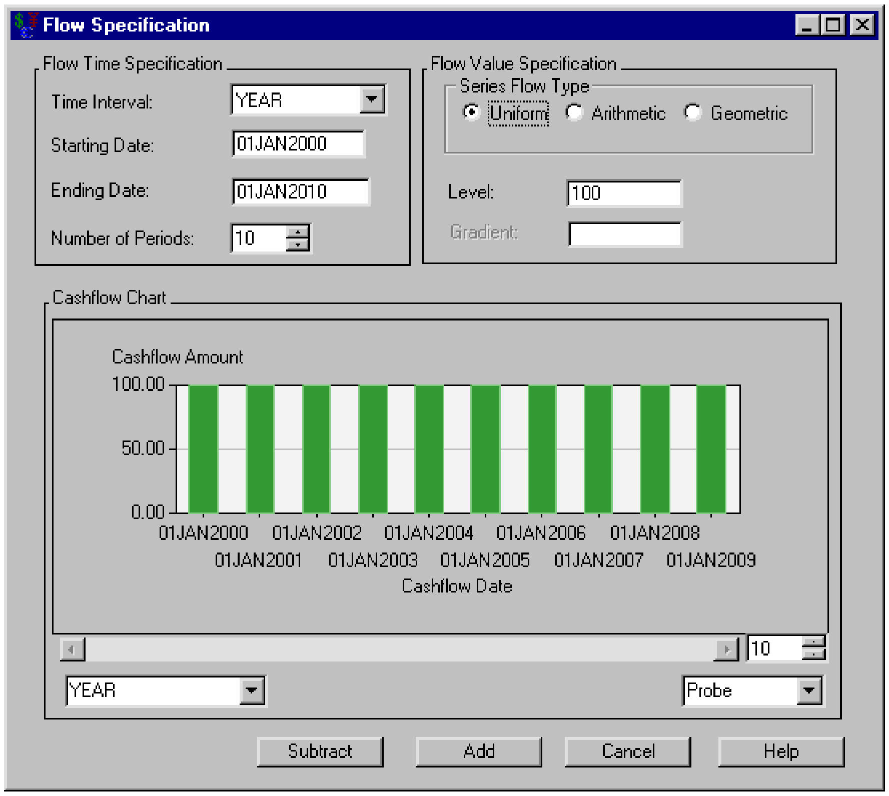
1. Right-click in the **Cashflow Specification** area and release on **Save...**. This opens the Save Dataset dialog box.
2. Enter the name of the SAS data set for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

Including a Generated Cashflow

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Specify...**. This opens the Flow Specification dialog box.
2. Select **YEAR** for the **Time Interval**.
3. Enter today’s date for the **Starting Date**.

4. Enter 10 for the **Number of Periods**. The **Ending Date** updates.
5. Enter 100 for the level. You can visualize the specification in the Cashflow Chart area (see Figure 44.19).
6. Click **Add** to add the specified cashflow to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.



Flow Specification

Flow Time Specification

Time Interval: YEAR

Starting Date: 01JAN2000

Ending Date: 01JAN2010

Number of Periods: 10

Flow Value Specification

Series Flow Type: [✓ Uniform] Arithmetic Geometric

Level: 100

Gradient:

Figure 44.19. Uniform Cashflow Specification

Clicking **Subtract** will subtract the current cashflow from the Generic Cashflow dialog box as it returns you to the Generic Cashflow dialog box.

You can generate arithmetic and geometric specifications by clicking them within the **Series Flow Type** area. However, you must enter a value for the **Gradient**. In both cases the **Level** value is the value of the list at the **Starting Date**. With an arithmetic flow type, entries increment by the value **Gradient** each **Time Interval**. With a geometric flow type, entries increase by the factor **Gradient** each **Time Interval**. Figure 44.20 displays an arithmetic cashflow with a **Level** of 100 and a **Gradient** of 12.

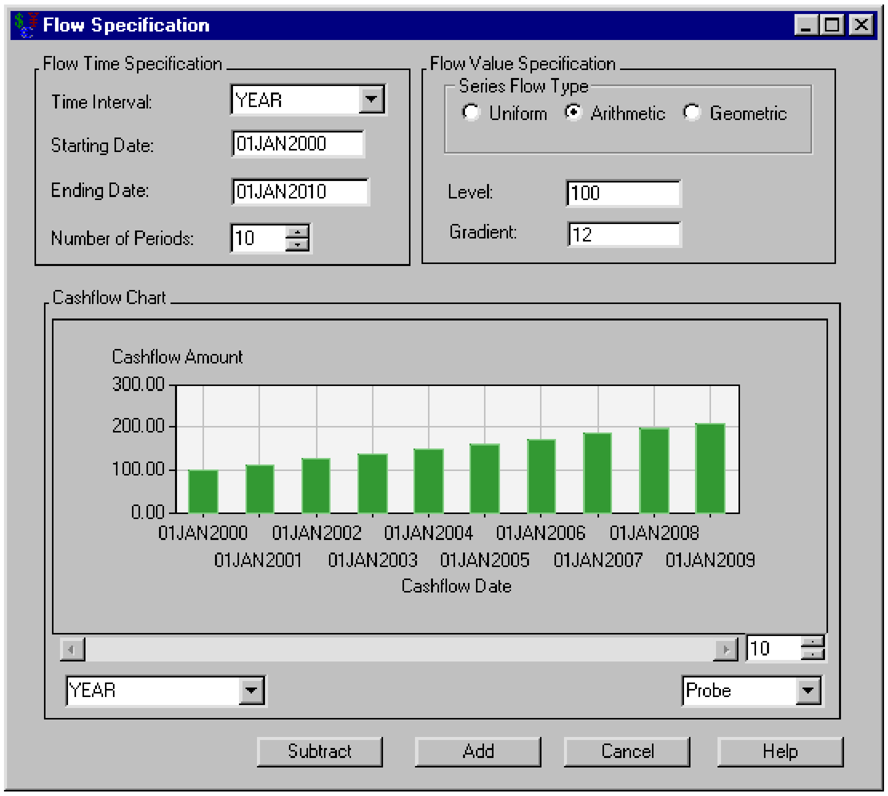


Figure 44.20. Arithmetic Cashflow Specification

Including a Forecasted Cashflow

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Forecast...**. This opens the **Forecast Specification** dialog box.
2. Enter SASHELP.RETAIL as the **Data Set**.
3. Select SALES for the **Analysis Variable**.
4. Click **Compute Forecast** to generate the forecast. You can visualize the forecast in the Cashflow Chart area (see Figure 44.21).
5. Click **Add** to add the forecast to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.

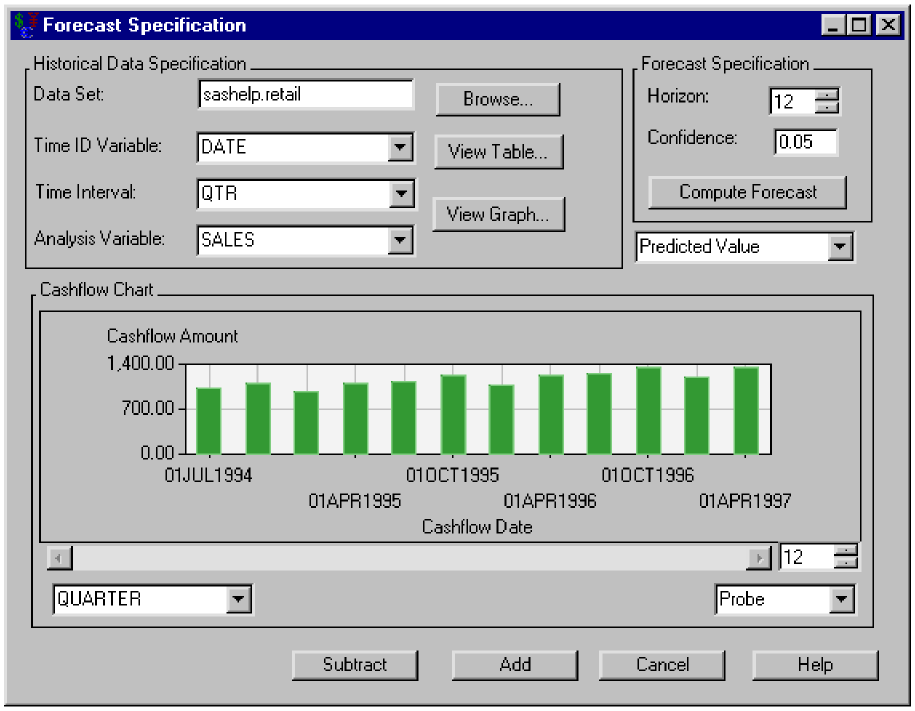


Figure 44.21. Cashflow Forecast

Clicking **Subtract** will subtract the current forecast from the Generic Cashflow dialog box as it returns you to the Generic Cashflow dialog box.

To review the values from the SAS data set you forecast, click **View Table...** or **View Graph...**.

You can adjust the following values for the SAS data set you forecast: **Time ID Variable**, **Time Interval**, and **Analysis Variable**.

You can adjust the following values for the forecast: the **Horizon**, the **Confidence**, and choice of predicted value, lower confidence limit, and upper confidence limit.

Using the Cashflow Chart

Three dialog boxes contain the Cashflow Chart to aide in your visualization of cashflows: Generic Cashflow, Flow Specification, and Forecast Specification. Within this chart, you possess the following tools:

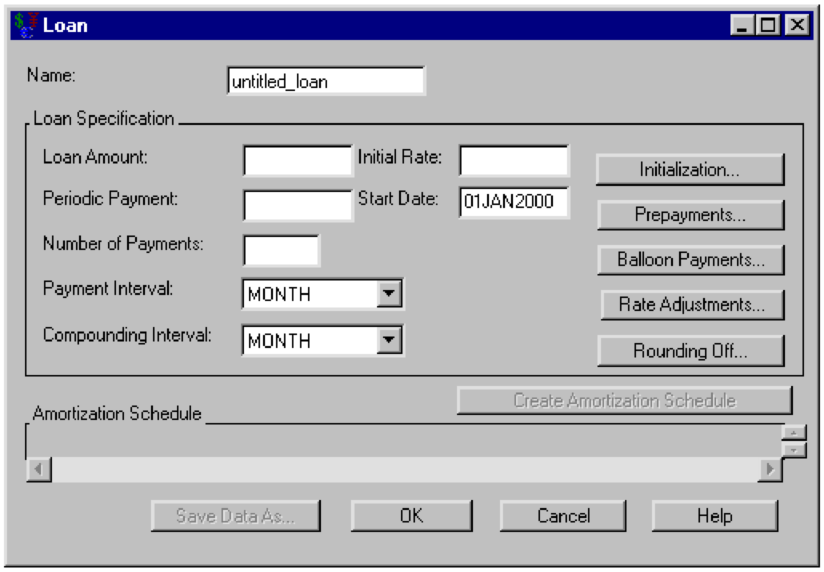
You can click on a bar in the plot and view its **Cashflow Date** and **Cashflow Amount**.

You can change the aggregation period of the view with the box in the lower left corner of the Cashflow Chart. You can take the quarterly sales figures from the previous example, select YEAR as the value for this box, and view the annual sales figures. You can change the number in the box to the right of the horizontal scroll bar to alter the number of entries you wish to view. The number in that box must be no greater than the number of entries in the cashflow list. Lessening this number has the effect of zooming in upon a portion of the cashflow. When the number is less than the number of entries in the cashflow list, you can use the scroll bar at the bottom of the chart to scroll through the chart.

Dialog Box Guide

Loan

Selecting **Investment** → **New** → **Loan...** from the Investment Analysis dialog box's menu bar opens the Loan dialog box displayed in Figure 44.22.



Name: untitled_loan

Start Date: 01JAN2000

Payment Interval: MONTH

Compounding Interval: MONTH

Figure 44.22. Loan Dialog Box

The following items are displayed:

Name holds the name you assign to the loan. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Loan Specification** area gives access to the values that define the loan.

Loan Amount holds the borrowed amount.

Periodic Payment holds the value of the periodic payments.

Number of Payments holds the number of payments in loan terms.

Payment Interval holds the frequency of the **Periodic Payment**.

Compounding Interval holds the compounding frequency.

Initial Rate holds the interest rate (a nominal percentage between 0 and 120) you pay on the loan.

Start Date holds the SAS date when the loan is initialized. The first payment is due one **Payment Interval** after this time.

Initialization... opens the Loan Initialization Options dialog box where you can define initialization costs and down-payments relevant to the loan.

Prepayments... opens the Loan Prepayments dialog box where you can specify the SAS dates and amounts of any prepayments.

Balloon Payments... opens the Balloon Payments dialog box where you can specify the SAS dates and amounts of any balloon payments.

Rate Adjustments... opens the Rate Adjustment Terms dialog box where you can specify terms for a variable-rate loan.

Rounding Off... opens the Rounding Off dialog box where you can select the number of decimal places for calculations.

Create Amortization Schedule becomes available when you adequately define the loan within the **Loan Specification** area. Clicking it generates the amortization schedule.

Amortization Schedule fills when you click **Create Amortization Schedule**. The schedule contains a row for the loan's start-date and each payment-date with information about the following:

Date is a SAS date, either the loan's start-date or a payment-date.

Beginning Principal Amount is the balance at that date.

Periodic Payment Amount is the expected payment at that date.

Interest Payment is zero for the loan's start-date; otherwise it holds the interest since the previous date.

Principal Repayment is the amount of the payment that went toward the principal.

Ending Principal is the balance at the end of the payment interval.

Print becomes available when you generate the amortization schedule. Clicking it sends the contents of the amortization schedule to the SAS session print device.

Save Data As... becomes available when you generate the amortization schedule. Clicking it opens the Save Output Dataset dialog box where you can save the amortization table (or portions thereof) as a SAS Dataset.

OK returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **OK** appends the current loan specification to the portfolio. If this is an existing loan specification, clicking **OK** returns the altered loan specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **Cancel** discards the current loan specification. If this is an existing loan specification, clicking **Cancel** discards the current editions.

Loan Initialization Options

Clicking **Initialization...** in the Loan dialog box opens the Loan Initialization Options dialog box displayed in Figure 44.23.

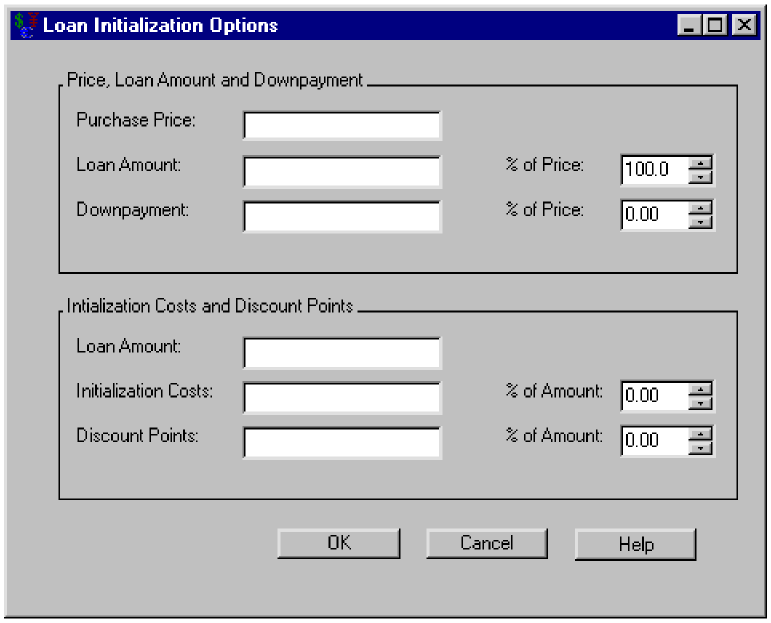


Figure 44.23. Loan Initialization Options Dialog Box

The following items are displayed:

The **Price, Loan Amount and Downpayment** area

> **Purchase Price** holds the actual price of the asset. This value equals the loan amount plus the downpayment.
>
> **Loan Amount** holds the loan amount.
>
> **% of Price** (to the right of **Loan Amount**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Loan Amount**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.
>
> **Downpayment** holds any downpayment paid for the asset.
>
> **% of Price** (to the right of **Downpayment**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Downpayment**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.

Initialization Costs and Discount Points area

Loan Amount holds a copy of the **Loan Amount** above.

Initialization Costs holds the value of any initialization costs.

% of Amount (to the right of **Initialization Costs**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Initialization Costs**. Setting the percentage manually causes the **Initialization Costs** to update.

Discount Points holds the value of any discount points.

% of Amount (to the right of **Discount Points**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Discount Points**. Setting the percentage manually causes the **Discount Points** to update.

OK returns you to the Loan dialog box, saving the information that is entered.

Cancel returns you to the Loan dialog box, discarding any editions since you opened the dialog box.

Loan Prepayments

Clicking **Prepayments...** in the Loan dialog box opens the Loan Prepayments dialog box displayed in Figure 44.24.

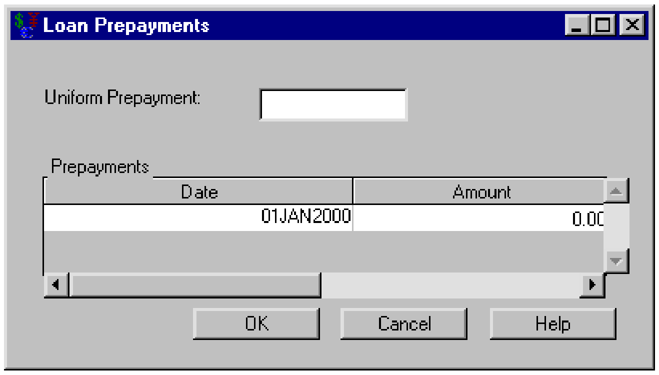


Figure 44.24. Loan Prepayments Dialog Box

The following items are displayed:

Uniform Prepayment holds the value of a regular prepayment concurrent to the usual periodic payment.

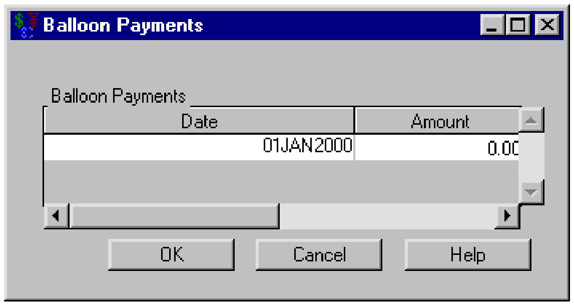
Prepayments holds a list of date-amount pairs to accommodate any prepayments. Right-clicking within the **Prepayments** area reveals many helpful tools for managing date-amount pairs.

OK returns you to the Loan dialog box, storing the information entered on the prepayments.

Cancel returns you to the Loan dialog box, discarding any prepayments entered since you opened the dialog box.

Balloon Payments

Clicking **Balloon Payments...** in the Loan dialog box opens the Balloon Payments dialog box displayed in Figure 44.25.



Date	Amount
01JAN2000	0.00

Figure 44.25. Balloon Payments Dialog Box

The following items are displayed:

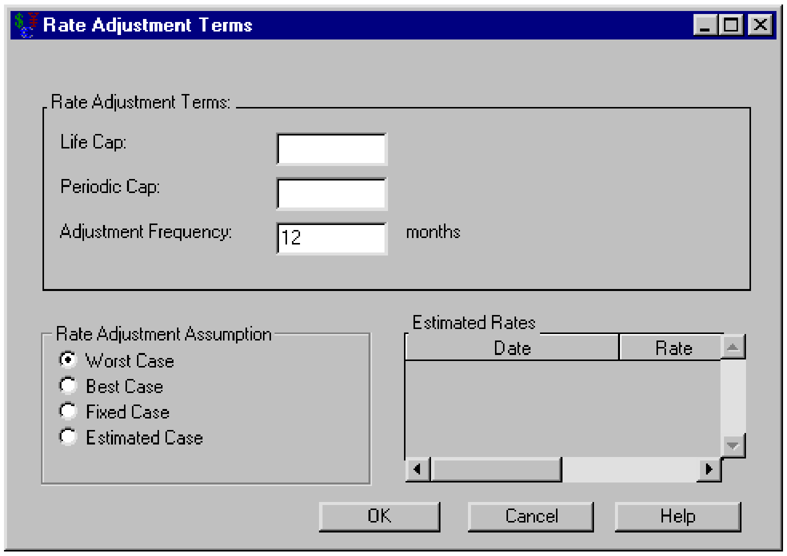
Balloon Payments holds a list of date-amount pairs to accommodate any balloon payments. Right-clicking within the **Balloon Payments** area reveals many helpful tools for managing date-amount pairs.

OK returns you to the Loan dialog box, storing the information entered on the balloon payments.

Cancel returns you to the Loan dialog box, discarding any balloon payments entered since you opened the dialog box.

Rate Adjustment Terms

Clicking **Rate Adjustments...** in the Loan dialog box opens the Rate Adjustment Terms dialog box displayed in Figure 44.26.



Rate Adjustment Terms

Rate Adjustment Terms:

Life Cap:

Periodic Cap:

Adjustment Frequency: 12 months

Rate Adjustment Assumption

[✓ Worst Case]
Best Case
Fixed Case
Estimated Case

Estimated Rates

Date	Rate

Figure 44.26. Rate Adjustment Terms Dialog Box

The following items are displayed:

The **Rate Adjustment Terms** area

> **Life Cap** holds the maximum deviation from the **Initial Rate** allowed over the life of the loan.
>
> **Periodic Cap** holds the maximum adjustment allowed per adjustment.
>
> **Adjustment Frequency** holds how often (in months) the lender can adjust the interest rate.

The **Rate Adjustment Assumption** determines the scenario the adjustments will take.

> **Worst Case** uses information from the **Rate Adjustment Terms** area to forecast a worst-case scenario.
>
> **Best Case** uses information from the **Rate Adjustment Terms** area to forecast a best-case scenario.
>
> **Fixed Case** specifies a fixed-rate loan.
>
> **Estimated Case** uses information from the **Rate Adjustment Terms** and **Estimated Rate** area to forecast a best-case scenario.

Estimated Rates holds a list of date-rate pairs, where each date is a SAS date and the rate is a nominal percentage between 0 and 120. The **Estimated Case** assumption uses these rates for its calculations. Right-clicking within the **Estimated Rates** area reveals many helpful tools for managing date-rate pairs.

OK returns you to the Loan dialog box, taking rate adjustment information into account.

Cancel returns you to the Loan dialog box, discarding any rate adjustment information provided since opening the dialog box.

Rounding Off

Clicking **Rounding Off...** in the Loan dialog box opens the Rounding Off dialog box displayed in Figure 44.27.

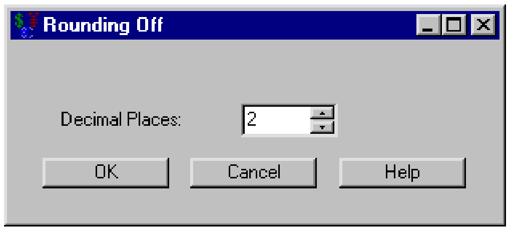


Figure 44.27. Rounding Off Dialog Box

The following items are displayed:

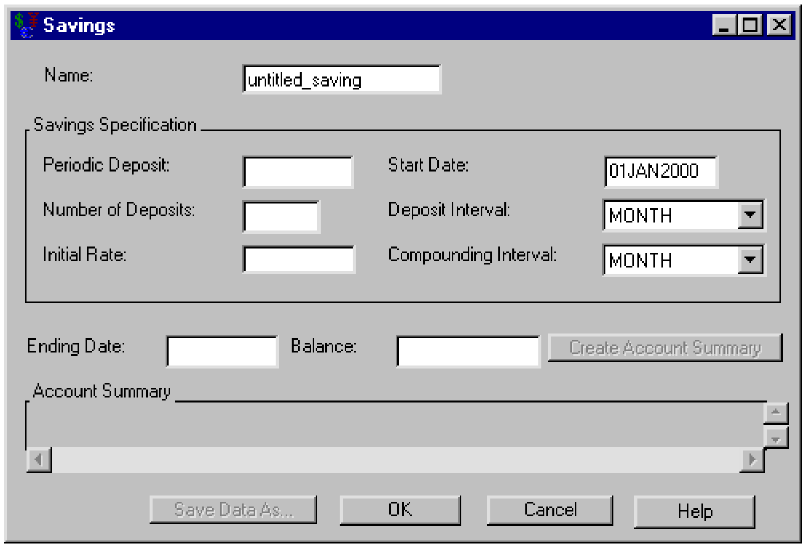
Decimal Places fixes the number of decimal places your results will display.

OK returns you to the Loan dialog box. Numeric values will then be represented with the number of decimals specified in **Decimal Places**.

Cancel returns you to the Loan dialog box. Numeric values will be represented with the number of decimals specified prior to opening this dialog box.

Savings

Selecting **Investment** → **New** → **Savings...** from the Investment Analysis dialog box's menu bar opens the Savings dialog box displayed in Figure 44.28.



Name: untitled_saving

Periodic Deposit:

Number of Deposits:

Initial Rate:

Start Date: 01JAN2000

Deposit Interval: MONTH

Compounding Interval: MONTH

Ending Date:

Balance:

Figure 44.28. Savings Dialog Box

The following items are displayed:

Name holds the name you assign to the savings. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Savings Specification** area

> **Periodic Deposit** holds the value of your regular deposits.
>
> **Number of Deposits** holds the number of deposits into the account.
>
> **Initial Rate** holds the interest rate (a nominal percentage between 0 and 120) the savings account earns.
>
> **Start Date** holds the SAS date when deposits begin.
>
> **Deposit Interval** holds the frequency of your **Periodic Deposit**.
>
> **Compounding Interval** holds how often the interest compounds.

Create Account Summary becomes available when you adequately define the savings within the **Savings Specification** area. Clicking it generates the account summary.

Account Summary fills when you click **Create Account Summary**. The schedule contains a row for each deposit-date with information about the following:

> **Date** is the SAS date of a deposit.
>
> **Starting Balance** is the balance at that date.
>
> **Deposits** is the deposit at that date.
>
> **Interest Earned** is the interest earned since the previous date.

Ending Balance is the balance after the payment.

Print becomes available when you generate an account summary. Clicking it sends the contents of the account summary to the SAS session print device.

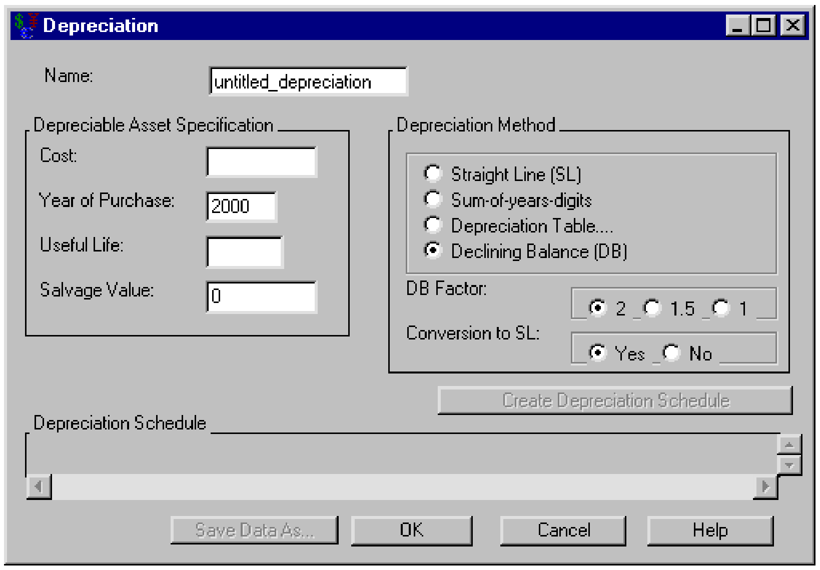
Save Data As... becomes available when you generate an account summary. Clicking it opens the Save Output Dataset dialog box where you can save the account summary (or portions thereof) as a SAS Dataset.

OK returns you to the Investment Analysis dialog box. If this is a new savings, clicking **OK** appends the current savings specification to the portfolio. If this is an existing savings specification, clicking **OK** returns the altered savings to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new savings, clicking **Cancel** discards the current savings specification. If this is an existing savings, clicking **Cancel** discards the current editions.

Depreciation

Selecting **Investment** → **New** → **Depreciation...** from the Investment Analysis dialog box's menu bar opens the Depreciation dialog box displayed in Figure 44.29.


Depreciation

Name: untitled_depreciation

Depreciable Asset Specification
Cost:
Year of Purchase: 2000
Useful Life:
Salvage Value: 0

Depreciation Method
Straight Line (SL)
Sum-of-years-digits
Depreciation Table....
[✓ Declining Balance (DB)]

DB Factor: [✓ 2] 1.5 1
Conversion to SL: [✓ Yes] No

Figure 44.29. Depreciation Dialog Box

The following items are displayed:

Name holds the name you assign to the depreciation. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Depreciable Asset Specification

Cost holds the asset's original cost.

Year of Purchase holds the asset's year of purchase.

Useful Life holds the asset's useful life (in years).

Salvage Value holds the asset's value at the end of its **Useful Life**.

The **Depreciation Method** area holds the depreciation methods available:

- Straight Line
- Sum-of-years Digits
- Depreciation Table...
- Declining Balance
  - DB Factor: choice of 2, 1.5, or 1
  - Conversion to SL: choice of Yes or No

Create Depreciation Schedule becomes available when you adequately define the depreciation within the **Depreciation Asset Specification** area. Clicking the **Create Depreciation Schedule** button then fills the **Depreciation Schedule** area.

Depreciation Schedule fills when you click **Create Depreciation Schedule**. The schedule contains a row for each year. Each row holds:

Year is a year

Start Book Value is the starting book value for that year

Depreciation is the depreciation value for that year

End Book Value is the ending book value for that year

Print becomes available when you generate the depreciation schedule. Clicking it sends the contents of the depreciation schedule to the SAS session print device.

Save Data As... becomes available when you generate the depreciation schedule. Clicking it opens the Save Output Dataset dialog box where you can save the depreciation table (or portions thereof) as a SAS Dataset.

OK returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **OK** appends the current depreciation specification to the portfolio. If this is an existing depreciation specification, clicking **OK** returns the altered depreciation specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **Cancel** discards the current depreciation specification. If this is an existing depreciation specification, clicking **Cancel** discards the current editions.

Depreciation Table

Clicking **Depreciation Table...** from **Depreciation Method** area of the Depreciation dialog box opens the Depreciation Table dialog box displayed in Figure 44.30.

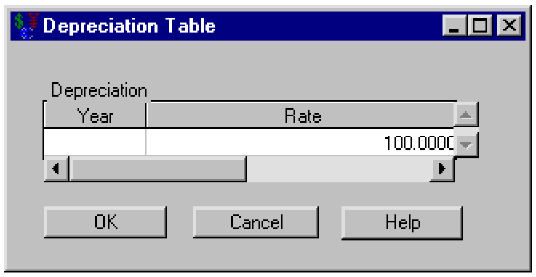


Figure 44.30. Depreciation Table Dialog Box

The following items are displayed:

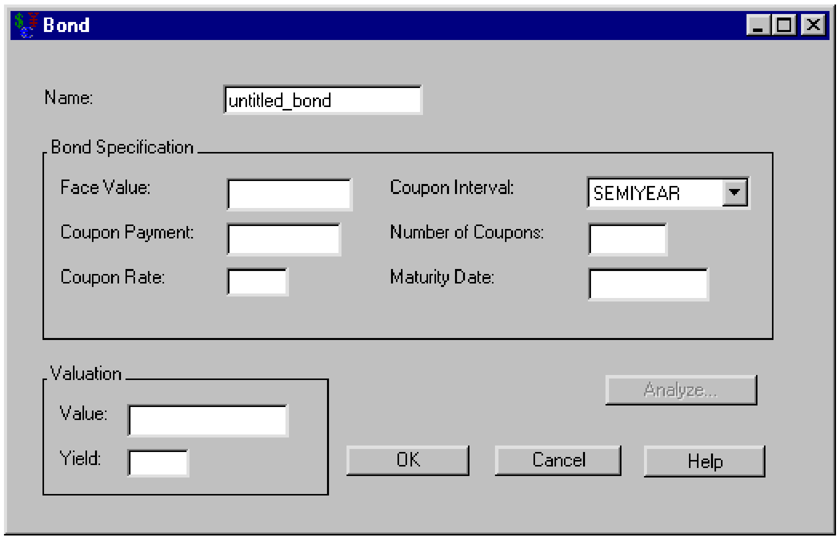
The **Depreciation** area holds a list of year-rate pairs where the rate is an annual depreciation rate (a percentage between 0% and 100%). Right-clicking within the **Depreciation** area reveals many helpful tools for managing year-rate pairs.

OK returns you to the Depreciation dialog box with the current list of depreciation rates from the **Depreciation** area.

Cancel returns you to the Depreciation dialog box, discarding any editions to the **Depreciation** area since you opened the dialog box.

Bond

Selecting **Investment** → **New** → **Bond...** from the Investment Analysis dialog box's menu bar opens the Bond dialog box displayed in Figure 44.31.



Name: untitled_bond

Bond Specification

Face Value:
Coupon Payment:
Coupon Rate:
Coupon Interval: SEMIYEAR
Number of Coupons:
Maturity Date:

Valuation

Value:
Yield:

Figure 44.31. Bond

The following items are displayed:

Name holds the name you assign to the bond. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Bond Specification

> **Face Value** holds the bond's value at maturity.
>
> **Coupon Payment** holds the amount of money you receive periodically as the bond matures.
>
> **Coupon Rate** holds the rate (a nominal percentage between 0% and 120%) of the **Face Value** that defines the **Coupon Payment**.
>
> **Coupon Interval** holds how often the bond pays its coupons.
>
> **Number of Coupons** holds the number of coupons before maturity.
>
> **Maturity Date** holds the SAS date when you can redeem the bond for its **Face Value**.

The **Valuation** area becomes available when you adequately define the bond within the **Bond Specification** area. Entering either the **Value** or the **Yield** causes the calculation of the other. If you respecify the bond after performing a calculation here, you must reenter the **Value** or **Yield** value to update the calculation.

> **Value** holds the bond's value if expecting the specified **Yield**.
>
> **Yield** holds the bond's yield if the bond is valued at the amount of **Value**.

You must specify the bond before analyzing it. Once you have specified the bond, clicking **Analyze...** opens the Bond Analysis dialog box where you can compare various values and yields.

OK returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **OK** appends the current bond specification to the portfolio. If this is an existing bond specification, clicking **OK** returns the altered bond specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **Cancel** discards the current bond specification. If this is an existing bond specification, clicking **Cancel** discards the current editions.

Bond Analysis

Clicking **Analyze...** from the Bond dialog box opens the Bond Analysis dialog box displayed in Figure 44.32.

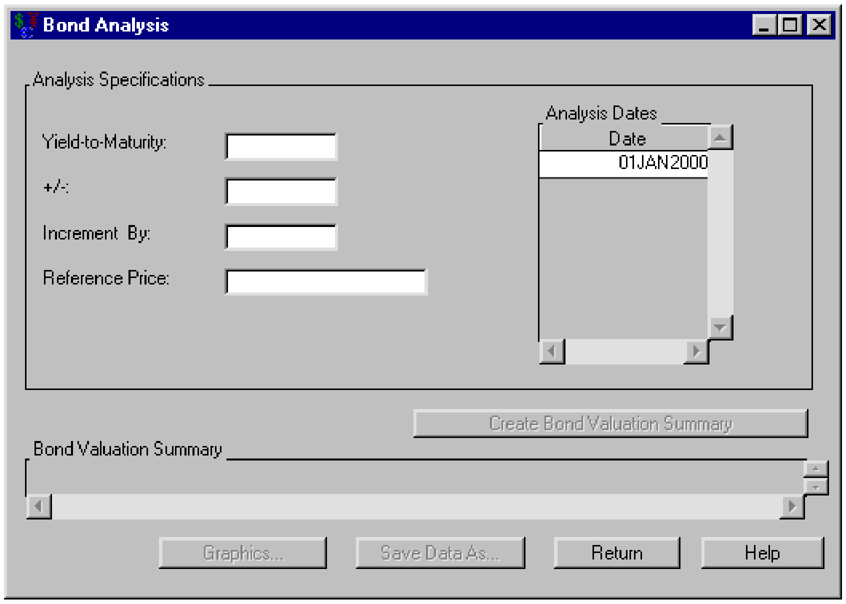


Figure 44.32. Bond Analysis

The following items are displayed:

Analysis Specifications

Yield-to-maturity holds the percentage yield upon which to center the analysis.

+/- holds the maximum deviation percentage from the **Yield-to-maturity** to consider.

Increment by holds the percentage increment by which the analysis is calculated.

Reference Price holds the reference price.

Analysis Dates holds a list of SAS dates for which you perform the bond analysis.

You must specify the analysis before valuing the bond for the various yields. Once you adequately specify the analysis, click **Create Bond Valuation Summary** to generate the bond valuation summary.

Bond Valuation Summary fills when you click **Create Bond Valuation Summary**. The schedule contains a row for each rate with information concerning the following:

Date is the SAS date when the **Value** gives the particular **Yield**.

Yield is the percent yield that corresponds to the **Value** at the given **Date**.

Value is the value of the bond at **Date** for the given **Yield**.

Percent Change is the percent change if the **Reference Price** is specified.

Duration is the duration.

Convexity is the convexity.

Graphics... opens the Bond Price graph representing the price versus yield-to-maturity.

Print becomes available when you generate the **Bond Valuation Summary**. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you fill the **Bond Valuation Summary** area. Clicking it opens the Save Output Dataset dialog box where you can save the valuation summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Bond dialog box.

Bond Price

Clicking **Graphics...** from the Bond dialog box opens the Bond Price dialog box displayed in Figure 44.33.

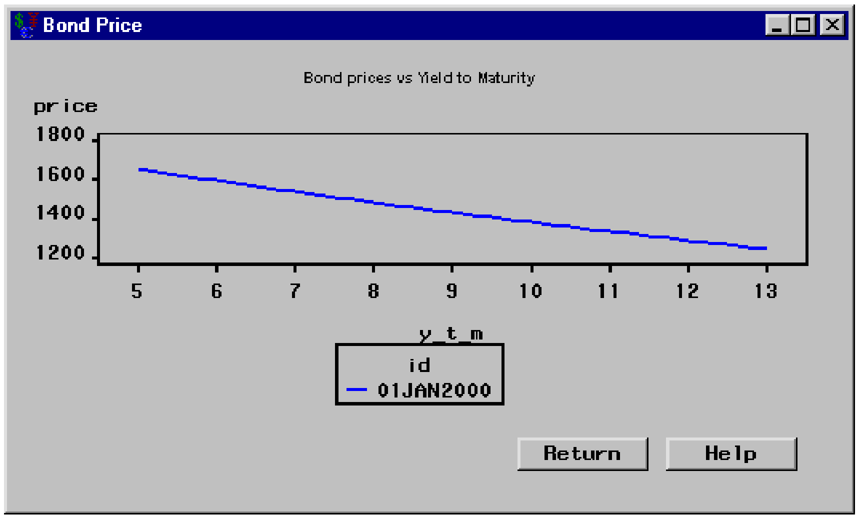


Figure 44.33. Bond Price Graph

It possesses the following item:

Return takes you back to the Bond Analysis dialog box.

Generic Cashflow

Selecting **Investment** → **New** → **Generic Cashflow...** from the Investment Analysis dialog box's menu bar opens the Generic Cashflow dialog box displayed in Figure 44.34.

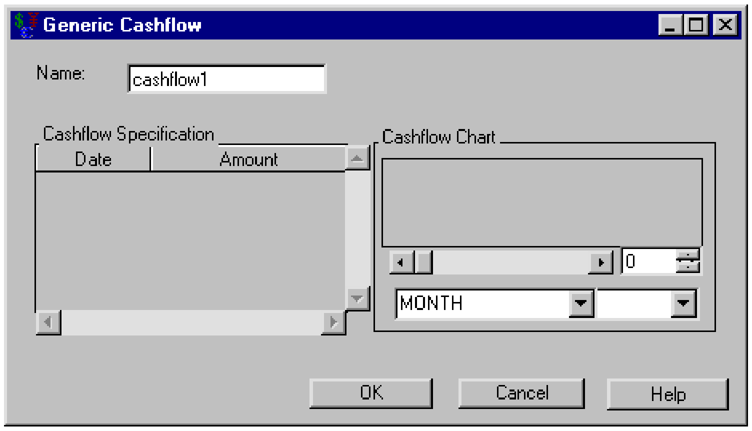

Name: cashflow1

Figure 44.34. Generic Cashflow

The following items are displayed:

Name holds the name you assign to the generic cashflow. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Cashflow Specification holds date-amount pairs corresponding to deposits and withdrawals (or benefits and costs) for the cashflow. Each date is a SAS date. Right-clicking within the **Cashflow Specification** area reveals many helpful tools for managing date-amount pairs.

The **Cashflow Chart** fills with a graph representing the cashflow when the **Cashflow Specification** area is nonempty. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

OK returns you to the Investment Analysis dialog box. If this is a new generic cashflow specification, clicking **OK** appends the current cashflow specification to the portfolio. If this is an existing cashflow specification, clicking **OK** returns the altered cashflow specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new cashflow specification, clicking **Cancel** discards the current cashflow specification. If this is an existing cashflow specification, clicking **Cancel** discards the current editions.

Right-Clicking within Generic Cashflow's Cashflow Specification Area

Right-click within the **Cashflow Specification** area of the Generic Cashflow dialog box pops up the menu displayed in Figure 44.35.

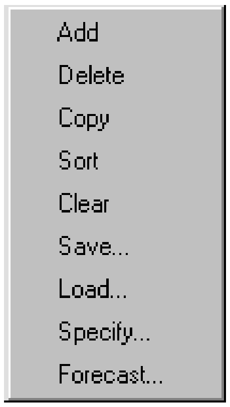


Figure 44.35. Right-Clicking

Add creates a blank pair.

Delete removes the currently highlighted pair.

Copy duplicates the currently selected pair.

Sort arranges the entered pairs in chronological order.

Clear empties the area of all pairs.

Save... opens the Save Dataset dialog box where you can save the entered pairs as a SAS Dataset for later use.

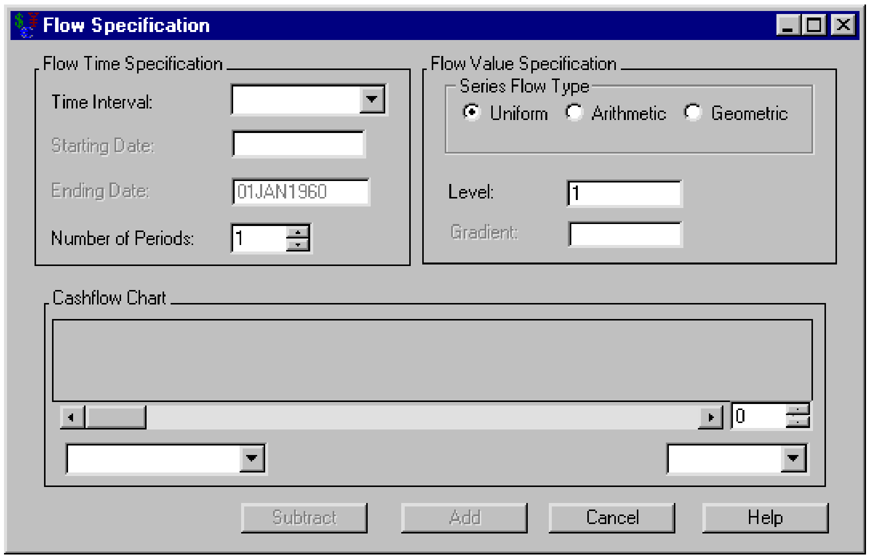
Load... opens the Load Dataset dialog box where you select a SAS Dataset to populate the area.

Specify... opens the Flow Specification dialog box where you can generate date-rate pairs to include in your cashflow.

Forecast... opens the Forecast Specification dialog box where you can generate the forecast of a SAS data set to include in your cashflow.

If you wish to perform one of these actions on a collection of pairs, you must select a collection of pairs before right-clicking. To select an adjacent list of pairs, do the following: click the first pair, hold down SHIFT, and click the final pair. Once the list of pairs is selected, you may release the SHIFT key.

Flow Specification


Flow Specification

Flow Time Specification
Time Interval:
Starting Date:
Ending Date: 01JAN1960
Number of Periods: 1

Flow Value Specification
Series Flow Type
[✓ Uniform] Arithmetic Geometric
Level: 1
Gradient:

Cashflow Chart
0

Figure 44.36. Flow Specification

The following items are displayed:

Flow Time Specification

Time Interval holds the uniform frequency of the entries.

You can set the **Starting Date** when you set the **Time Interval**. It holds the SAS date the entries will start.

You can set the **Ending Date** when you set the **Time Interval**. It holds the SAS date the entries will end.

Number of Periods holds the number of entries.

Flow Value Specification

Series Flow Type describes the movement the entries can assume:

- **Uniform** assumes all entries are equal.
- **Arithmetic** assumes the entries begin at **Level** and increase by the value of **Gradient** per entry.
- **Geometric** assumes the entries begin at **Level** and increase by a factor of **Gradient** per entry.

Level holds the starting amount for all **Flow Type**s.

You can set the **Gradient** when you select either **Arithmetic** or **Geometric Gradient**. It holds the arithmetic and geometric gradients, respectively, for the **Arithmetic** and **Geometric Flow Type**s.

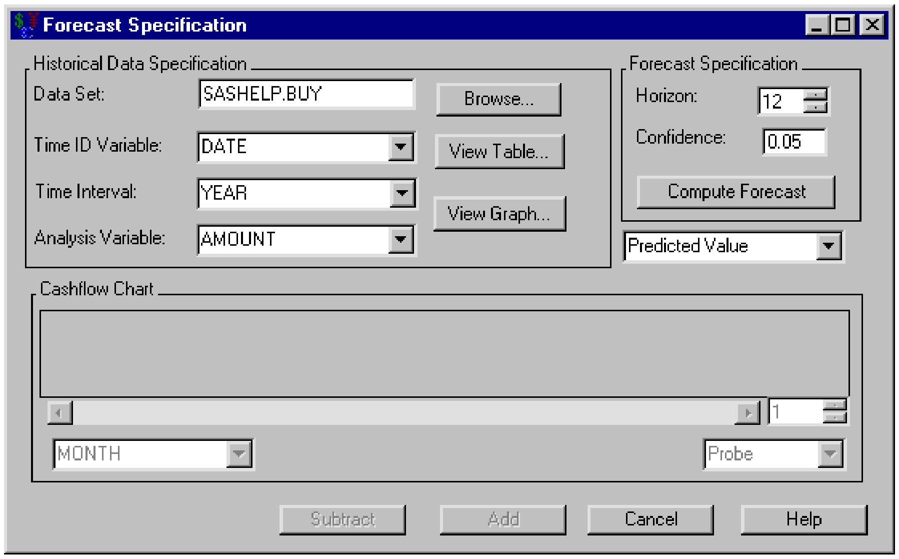
The **Cashflow Chart** fills with a graph displaying the dates and values of the entries when the cashflow entries are adequately defined. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

Subtract becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box subtracting the entries from the current cashflow.

Add becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow dialog box adding the entries to the current cashflow.

Cancel returns you to Generic Cashflow dialog box without editing the cashflow.

Forecast Specification



Forecast Specification

Historical Data Specification

Data Set: SASHELP.BUY

Time ID Variable: DATE

Time Interval: YEAR

Analysis Variable: AMOUNT

Forecast Specification

Horizon: 12

Confidence: 0.05

Predicted Value

Cashflow Chart

1

MONTH

Probe

Figure 44.37. Forecast Specification

The following items are displayed:

Historical Data Specification

Data Set holds the name of the SAS data set to forecast.

Browse... opens the standard SAS **Open** dialog box to help select a SAS data set to forecast.

Time ID Variable holds the time ID variable to forecast over.

Time Interval fixes the time interval for the **Time ID Variable**.

Analysis Variable holds the data variable upon which to forecast.

View Table... opens a table that displays in a list the contents of the specified SAS data set.

View Graph... opens the Time Series Viewer that graphically displays the contents of the specified SAS data set.

Forecast Specification

Horizon holds the number of periods into the future you wish to forecast.

Confidence holds the confidence limit for applicable forecasts.

Compute Forecast fills the **Cashflow Chart** with the forecast.

The following box holds the type of forecast you wish to generate:

- Predicted Value

- Lower Confidence Limit
- Upper Confidence Limit

The **Cashflow Chart** fills when you click **Compute Forecast**. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

Subtract becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box subtracting the forecast from the current cashflow.

Add becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow adding the forecast to the current cashflow.

Cancel returns to Generic Cashflow dialog box without editing the cashflow.

Chapter 45
Computations

Chapter Contents

Chapter 45
Computations

The Compute Menu

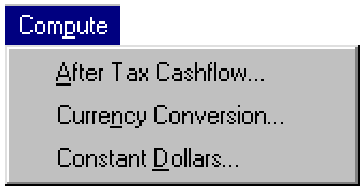


Figure 45.1. The Compute Menu

The **Compute** menu, shown in Figure 45.1, offers the following options that apply to generic cashflows.

After Tax Cashflow opens the After Tax Cashflow Calculation dialog box. Computing an after tax cashflow is useful when taxes affect investment alternatives differently. Comparing after tax cashflows provides a more accurate determination of the cashflows' profitabilities. You can set default values for income tax rates by selecting **Tools** → **Define Rate** → **Income Tax Rate...** from the Investment Analysis dialog box. This opens the Income Tax Specification dialog box where you can enter the tax rates.

Currency Conversion opens the Currency Conversion dialog box. Currency conversion is necessary when investments are in different currencies. For data concerning currency conversion rates, see http://dsbb.imf.org/, the International Monetary Fund's Dissemination Standards Bulletin Board.

Constant Dollars opens the Constant Dollar Calculation dialog box. A constant dollar (inflation adjusted monetary value) calculation takes cashflow and inflation information and discounts the cashflow to a level where the buying power of the monetary unit is "constant" over time. Groups quantify inflation (in the form of price indices and inflation rates) for countries and industries by averaging the growth of prices for various products and sectors of the economy. For data concerning price indices, see the United States Department of Labor at http://www.dol.gov/ and the International Monetary Fund's Dissemination Standards Bulletin Board at http://dsbb.imf.org/. You can set default values for inflation rates by clicking **Tools** → **Define Rate** → **Inflation...** from the Investment Analysis dialog box. This opens the Inflation Specification dialog box where you can enter the inflation rates.

Tasks

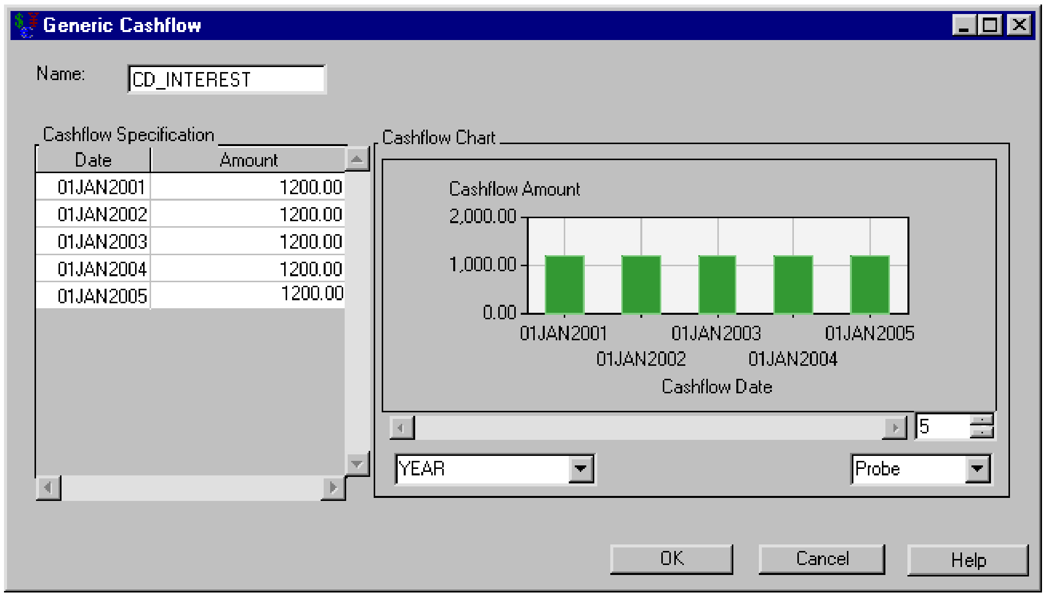
The next few sections show how to perform computations for the following situation. Suppose you buy a $10,000 certificate of deposit that pays 12% interest a year for five years. Your earnings are taxed at a rate of 30% federally and 7% locally. Also, you want to transfer all the money to an account in England. British pounds convert to American dollars at an exchange rate of $1.00 to £0.60. The inflation rate in England is 3%. The instructions in this example assume familiarity with the following:

- The right-clicking options of the **Cashflow Specification** area in the Generic Cashflow dialog box (described in "Right-Clicking within Generic Cashflow's Cashflow Specification Area" on page 2306.)
- The **Save Data As...** button located in many dialog boxes (described in "Saving Output to SAS Datasets" on page 2343.)

Taxing a Cashflow

Consider the example described in "The Compute Menu" on page 2313. To create the earnings, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow** to create a generic cashflow.
2. Enter CD_INTEREST for the **Name**.
3. Enter 1200 for each of the five years starting one year from today as displayed in Figure 45.2.
4. Click **OK** to return to the Investment Analysis dialog box.



Name: CD_INTEREST

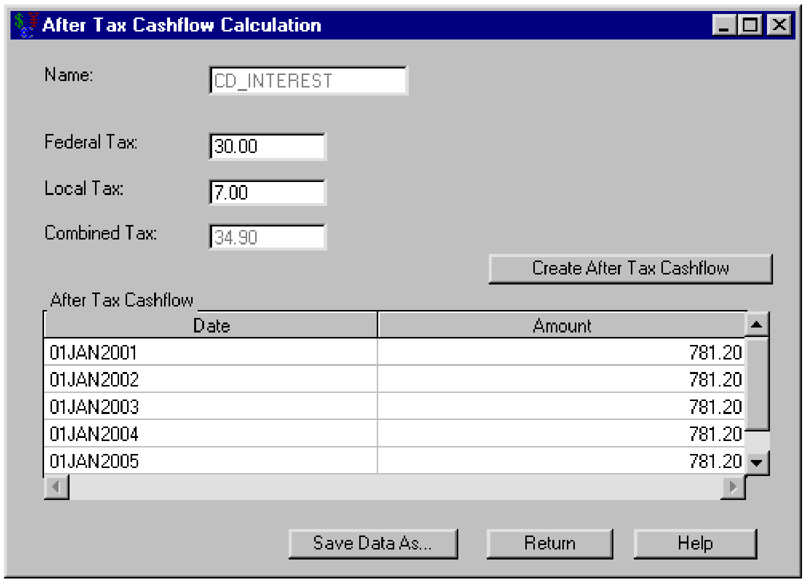
Cashflow Specification

Date	Amount
01JAN2001	1200.00
01JAN2002	1200.00
01JAN2003	1200.00
01JAN2004	1200.00
01JAN2005	1200.00

Figure 45.2. Computing the Interest on the CD

To compute the tax on the earnings, follow these steps:

1. Select CD_INTEREST from the **Portfolio** area.
2. Select **Compute** → **After Tax Cashflow** from the pull-down menu.
3. Enter 30 for **Federal Tax**.
4. Enter 7 for **Local Tax**. Note that **Combined Tax** updates.
5. Click **Create After Tax Cashflow** and the **After Tax Cashflow** area fills, as displayed in Figure 45.3.



Name: CD_INTEREST

Federal Tax: 30.00

Local Tax: 7.00

Combined Tax: 34.90

After Tax Cashflow

Date	Amount
01JAN2001	781.20
01JAN2002	781.20
01JAN2003	781.20
01JAN2004	781.20
01JAN2005	781.20

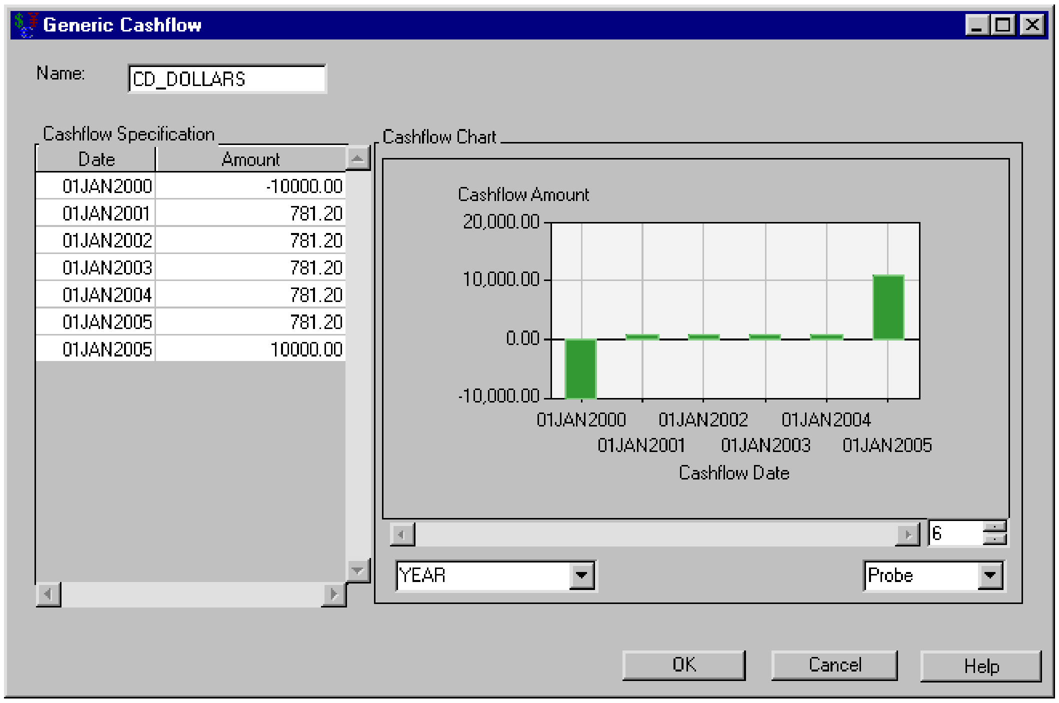
Figure 45.3. Computing the Interest After Taxes

Save the taxed earnings to a SAS data set named WORK.CD_AFTERTAX. Click **Return** to return to the Investment Analysis dialog box.

Converting Currency

Consider the example described in "The Compute Menu" on page 2313. To create the cashflow to convert, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow...** to open a new generic cashflow.
2. Enter CD_DOLLARS for the **Name**.
3. Load WORK.CD_AFTERTAX into its **Cashflow Specification**.
4. Add -10,000 for today and +10,000 for five years from today to the cashflow as displayed in Figure 45.4.
5. Sort the transactions by date to aid your reading.
6. Click **OK** to return to the Investment Analysis dialog box.



Name: CD_DOLLARS

Cashflow Specification

Date	Amount
01JAN2000	-10000.00
01JAN2001	781.20
01JAN2002	781.20
01JAN2003	781.20
01JAN2004	781.20
01JAN2005	781.20
01JAN2005	10000.00

6

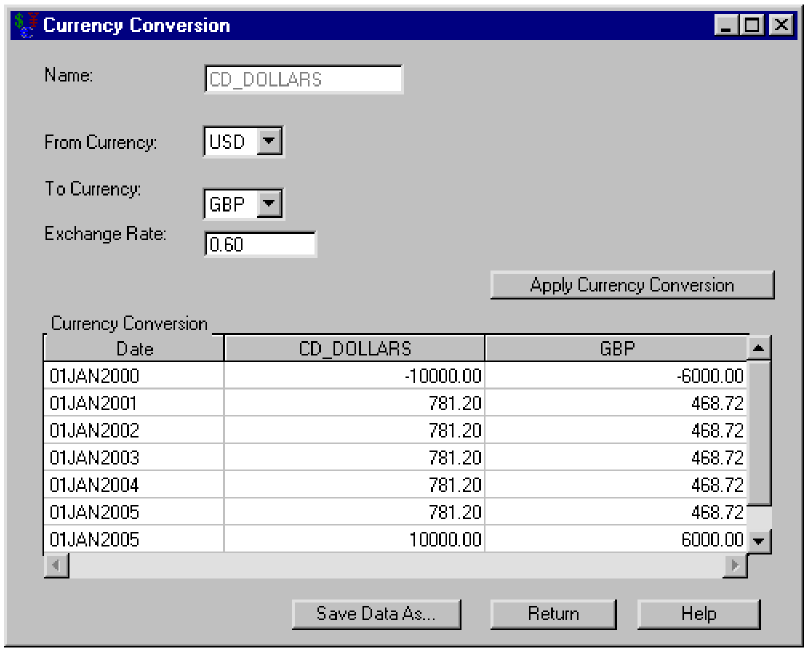
YEAR

Probe

Figure 45.4. The CD in Dollars

To convert from British pounds to American dollars, follow these steps:

1. Select CD_DOLLARS from the portfolio.
2. Select **Compute** → **Currency Conversion** from the pull-down menu. This opens the Currency Conversion dialog box.
3. Select USD for the **From Currency**.
4. Select GBP for the **To Currency**.
5. Enter 0.60 for the **Exchange Rate**.
6. Click **Apply Currency Conversion** to fill the **Currency Conversion** area as displayed in Figure 45.5.



Name: CD_DOLLARS

From Currency: USD

To Currency: GBP

Exchange Rate: 0.60

Currency Conversion

Date	CD_DOLLARS	GBP
01JAN2000	-10000.00	-6000.00
01JAN2001	781.20	468.72
01JAN2002	781.20	468.72
01JAN2003	781.20	468.72
01JAN2004	781.20	468.72
01JAN2005	781.20	468.72
01JAN2005	10000.00	6000.00

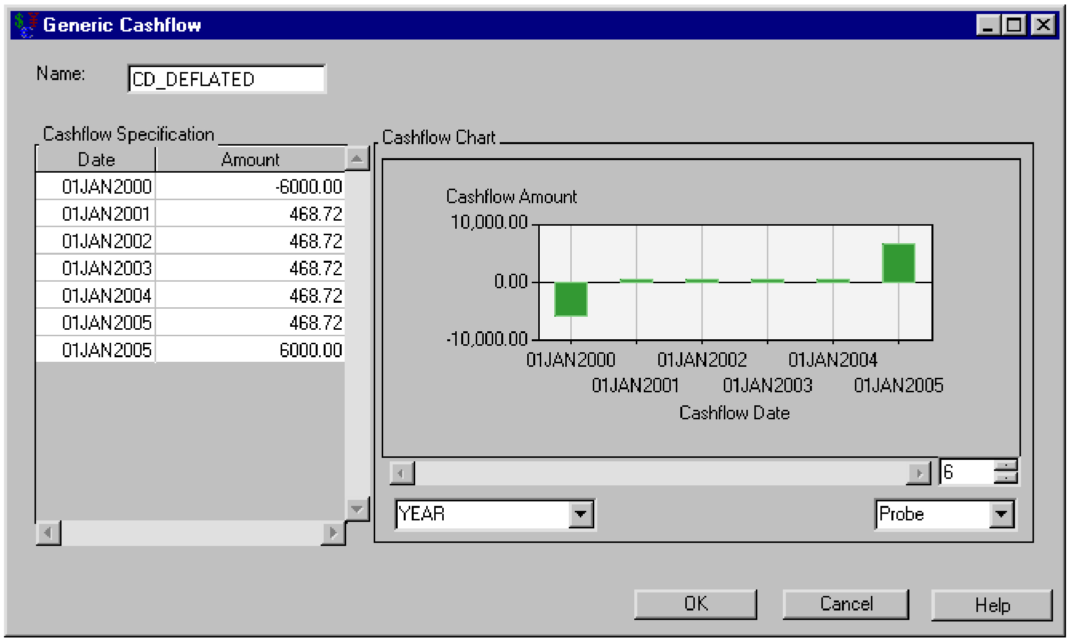
Figure 45.5. Converting the CD to Pounds

Save the converted values to a SAS data set named WORK.CD_POUNDS. Click **Return** to return to the Investment Analysis dialog box.

Inflating Cashflows

Consider the example described in "The Compute Menu" on page 2313. To create the cashflow to deflate, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow...** to open a new generic cashflow.
2. Enter CD_DEFLATED for **Name**.
3. Load WORK.CD_POUNDS into its **Cashflow Specification** (see Figure 45.6).
4. Click **OK** to return to the Investment Analysis dialog box.



Name: CD_DEFLATED

Cashflow Specification

Date	Amount
01JAN2000	-6000.00
01JAN2001	468.72
01JAN2002	468.72
01JAN2003	468.72
01JAN2004	468.72
01JAN2005	468.72
01JAN2005	6000.00

6

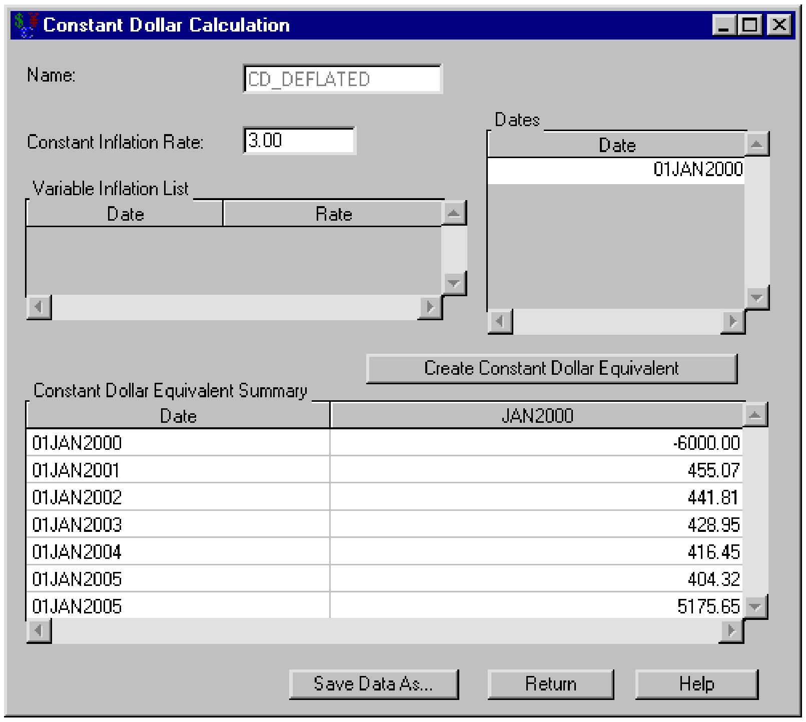
YEAR

Probe

Figure 45.6. The CD before Deflation

To deflate the values, follow these steps:

1. Select CD_DEFLATED from the portfolio.
2. Select **Compute** → **Constant Dollars** from the pull-down menu. This opens the Constant Dollar Calculation dialog box.
3. Clear the **Variable Inflation List** area.
4. Enter 3 for the **Constant Inflation Rate**.
5. Click **Create Constant Dollar Equivalent** to generate a constant dollar equivalent summary (see Figure 45.7).



Variable Inflation List

Date	Rate

Dates

Date
01JAN2000

Constant Dollar Equivalent Summary

Date	JAN2000
01JAN2000	-6000.00
01JAN2001	455.07
01JAN2002	441.81
01JAN2003	428.95
01JAN2004	416.45
01JAN2005	404.32
01JAN2005	5175.65

Figure 45.7. CD Values after Deflation

You can save the deflated cashflow to a SAS data set for use in an internal rate of return analysis or breakeven analysis.

Click **Return** to return to the Investment Analysis dialog box.

Dialog Box Guide

After Tax Cashflow Calculation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an after tax calculation, select **Compute** → **After Tax...** from the Investment Analysis dialog box's menu bar. This opens the After Tax Cashflow Calculation dialog box displayed in Figure 45.8.

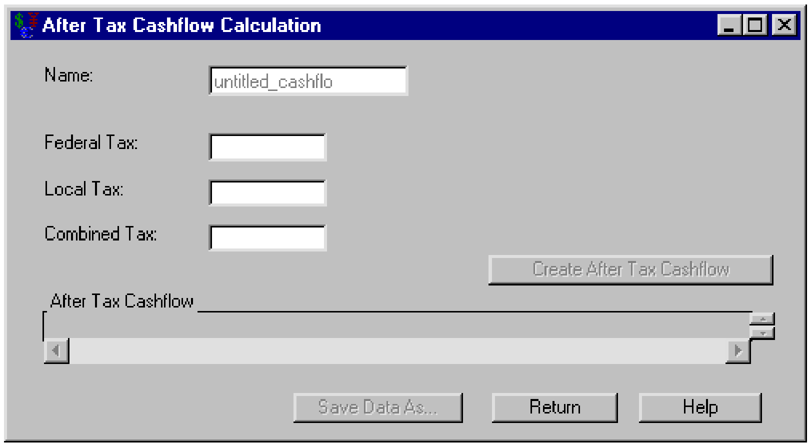


Figure 45.8. After Tax Cashflow Calculation Dialog Box

The following items are displayed:

Name holds the name of the investment for which you are computing the after tax cashflow.

Federal Tax holds the federal tax rate (a percentage between 0% and 100%).

Local Tax holds the local tax rate (a percentage between 0% and 100%).

Combined Tax holds the effective tax rate from federal and local income taxes.

Create After Tax Cashflow becomes available when **Combined Tax** is non-empty. Clicking **Create After Tax Cashflow** then fills the **After Tax Cashflow** area.

After Tax Cashflow fills when you click **Create After Tax Cashflow**. It holds a list of date-amount pairs where the amount is the amount retained after taxes for that date.

Print becomes available when you fill the after tax cashflow. Clicking it sends the contents of the after tax cashflow to the SAS session print device.

Save Data As... becomes available when you fill the after tax cashflow. Clicking it opens the Save Output Dataset dialog box where you can save the resulting cashflow (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Currency Conversion

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a currency conversion, select **Compute** → **Currency Conversion...** from the Investment Analysis dialog box's menu bar. This opens the Currency Conversion dialog box displayed in Figure 45.9.

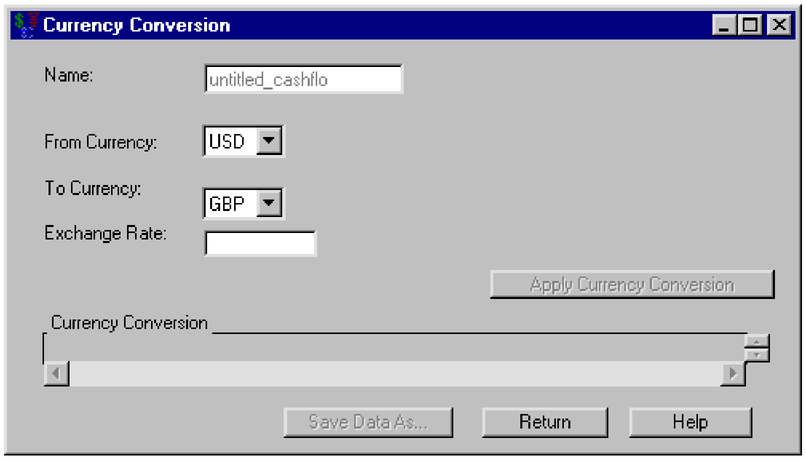


Figure 45.9. Currency Conversion Dialog Box

The following items are displayed:

Name holds the name of the investment to which you are applying the currency conversion.

From Currency holds the name of the currency the cashflow currently represents.

To Currency holds the name of the currency to which you wish to convert.

Exchange Rate holds the rate of exchange between the **From Currency** and the **To Currency**.

Apply Currency Conversion becomes available when you fill **Exchange Rate**. Clicking **Apply Currency Conversion** fills the **Currency Conversion** area.

Currency Conversion fills when you click **Apply Currency Conversion**. The schedule contains a row for each cashflow item with the following information:

- **Date** is a SAS date within the cashflow.
- The **From Currency** value is the amount in the original currency at that date.
- The **To Currency** value is the amount in the new currency at that date.

Print becomes available when you fill the **Currency Conversion** area. Clicking it sends the contents of the conversion table to the SAS session print device.

Save Data As... becomes available when you fill the **Currency Conversion** area. Clicking it opens the Save Output Dataset dialog box where you can save the conversion table (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Constant Dollar Calculation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a constant dollar calculation, select **Compute** → **Constant Dollars...** from the Investment Analysis dialog box's menu bar. This opens the Constant Dollar Calculation dialog box displayed in Figure 45.10.

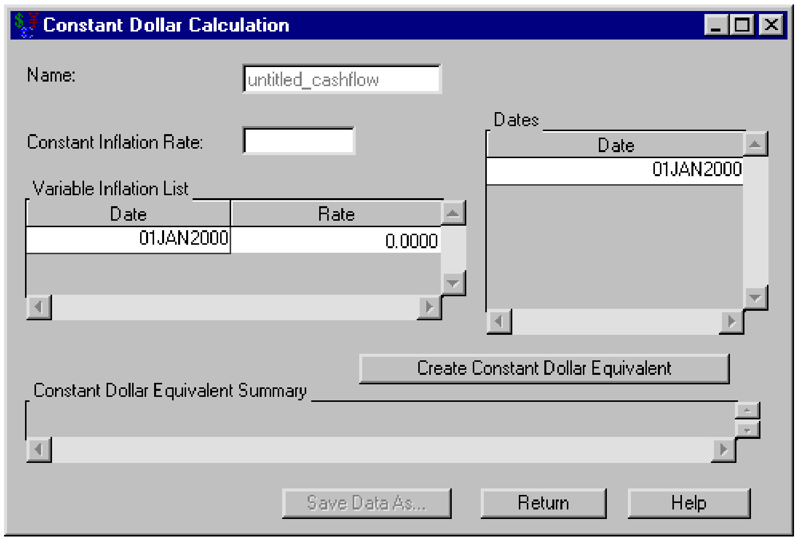


Figure 45.10. Constant Dollar Calculation Dialog Box

The following items are displayed:

Name holds the name of the investment for which you are computing the constant dollars value.

Constant Inflation Rate holds the constant inflation rate (a percentage between 0% and 120%). This value is used if the **Variable Inflation List** area is empty.

Variable Inflation List holds date-rate pairs that describe how inflation varies over time. Each date is a SAS date, and the rate is a percentage between 0% and 120%. Each date refers to when that inflation rate begins. Right-clicking within the **Variable Inflation** area reveals many helpful tools for managing date-rate pairs. If you assume a fixed inflation rate, just insert that rate in **Constant Rate**.

Dates holds the SAS date(s) at which you wish to compute the constant dollar equivalent. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

Create Constant Dollar Equivalent becomes available when you enter inflation rate information. Clicking it fills the constant dollar equivalent summary with the computed constant dollar values.

Constant Dollar Equivalent Summary fills with a summary when you click **Create Constant Dollar Equivalent**. The first column lists the dates of the generic cashflow. The second column contains the constant dollar equivalent of the original generic cashflow item of that date.

Print becomes available when you fill the constant dollar equivalent summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you fill the constant dollar equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Chapter 46
Analyses

Chapter Contents

Chapter 46
Analyses

The Analyze Menu

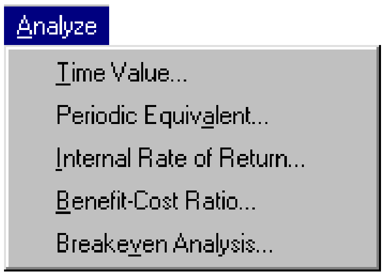


Figure 46.1. Analyze Menu

The **Analyze** menu, shown in Figure 46.1, offers the following options for use on applicable investments.

Time Value opens the Time Value Analysis dialog box. Time value analysis involves moving money through time across a defined MARR so that you can compare value at a consistent date. The MARR can be constant or variable over time.

Periodic Equivalent opens the Uniform Periodic Equivalent dialog box. Uniform periodic equivalent analysis determines the payment needed to convert a cashflow to uniform amounts over time, given a periodicity, a number of periods, and a MARR. This option helps when making comparisons where one alternative is uniform (such as renting) and another is not (such as buying).

Internal Rate of Return opens the Internal Rate of Return dialog box. The internal rate of return of a cashflow is the interest rate that makes the time value equal to 0. This calculation assumes uniform periodicity of the cashflow. It is particularly applicable where the choice of MARR would be difficult.

Benefit-Cost Ratio opens the Benefit-Cost Ratio Analysis dialog box. The benefit-cost ratio divides the time value of the benefits by the time value of the costs. For example, governments often use this analysis when deciding whether to commit to a public works project.

Breakeven Analysis opens the Breakeven Analysis dialog box. Breakeven analysis computes time values at various MARRs to compare, which can be advantageous when it is difficult to determine a MARR. This analysis can help you determine how the cashflow's profitability varies with your choice of MARR. A graph displaying the relationships between time value and MARR is also available.

Tasks

Performing Time Value Analysis

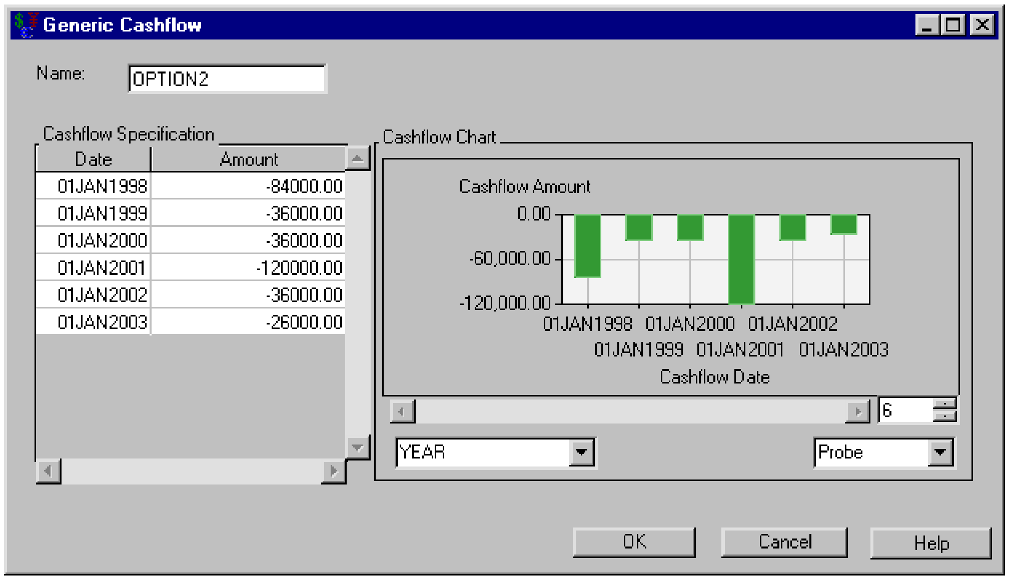
Suppose a rock quarry needs equipment to use the next five years. It has two alternatives:

- A box loader and conveyer system that has a one-time cost of $264,000.
- A two-shovel loader which costs $84,000 but has a yearly operating cost of $36,000. This loader has a service life of three years, which necessitates the purchase of a new loader for the final two years of the rock quarry project. Assume the second loader also costs $84,000 and its salvage value after its two-year service is $10,000. A SAS data set that describes this is available at SASHELP.ROCKPIT.

You expect a 13% MARR. Which is the better alternative?

To create the cashflows, follow these steps:

1. Create a cashflow with the single amount -264,000. Date the amount 01JAN1998 to be consistent with the SAS data set you load.
2. Load SASHELP.ROCKPIT into a second cashflow, as displayed in Figure 46.2.



Name: OPTION2

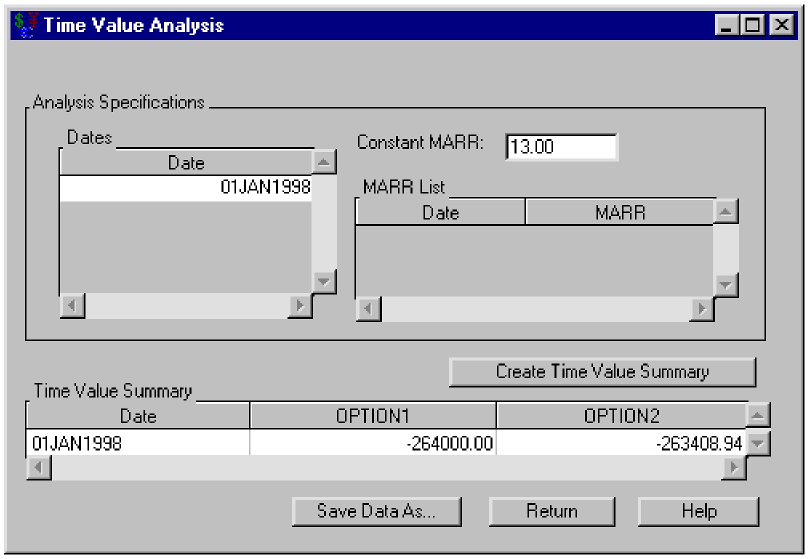
Date	Amount
01JAN1998	-84000.00
01JAN1999	-36000.00
01JAN2000	-36000.00
01JAN2001	-120000.00
01JAN2002	-36000.00
01JAN2003	-26000.00

Figure 46.2. The contents of SASHELP.ROCKPIT

To compute the time values of these investments, follow these steps:

1. Select both cashflows.

2. Select **Analyze** → **Time Value...**. This opens the Time Value Analysis dialog box.
3. Enter the date 01JAN1998 into the **Dates** area.
4. Enter 13 for the **Constant MARR**.
5. Click **Create Time Value Summary**.



Constant MARR: 13.00

Date
01JAN1998

MARR List

Date	MARR

Time Value Summary

Date	OPTION1	OPTION2
01JAN1998	-264000.00	-263408.94

Figure 46.3. Performing the Time Value Analysis

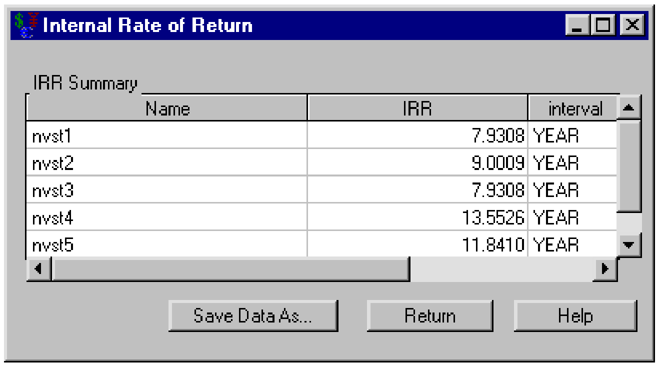
As shown in Figure 46.3, option 1 has a time value of -$264,000.00 naturally on 01JAN1998. However, option 2 has a time value of -$263,408.94, which is slightly less expensive.

Computing an Internal Rate of Return

You are choosing between five investments. A portfolio containing these investments is available at SASHELP.INVSAMP.NVST. Which investments are acceptable if you expect a MARR of 9%?

Open the portfolio SASHELP.INVSAMP.NVST and compare the investments. Note that Internal Rate of Return computations assume regular periodicity of the cashflow. To compute the internal rates of return, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Internal Rate of Return...**.



IRR Summary

Name	IRR	interval
nvst1	7.9308	YEAR
nvst2	9.0009	YEAR
nvst3	7.9308	YEAR
nvst4	13.5526	YEAR
nvst5	11.8410	YEAR

Figure 46.4. Computing an Internal Rate of Return

The results displayed in Figure 46.4 indicate that the internal rates of return for investments 2, 4, and 5 are greater than 9%. Hence, each of these is acceptable.

Performing a Benefit-Cost Ratio Analysis

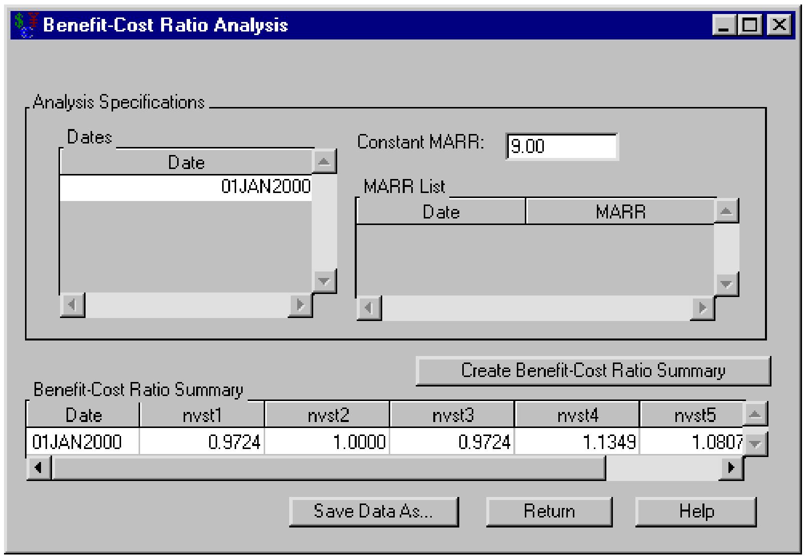
Suppose a municipality has excess funds to invest. It is choosing between the same investments described in the previous example. Government agencies often compute benefit-cost ratios to decide which investment to pursue. Which is best in this case?

Open the portfolio SASHELP.INVSAMP.NVST and compare the investments.

To compute the benefit-cost ratios, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Benefit-Cost Ratio...**.
3. Enter 01JAN1996 for the **Date**.
4. Enter 9 for **Constant MARR**.
5. Click **Create Benefit-Cost Ratio Summary** to fill the **Benefit-Cost Ratio Summary** area.

The results displayed in Figure 46.5 indicate that investments 2, 4, and 5 have ratios greater than 1. Therefore, each is profitable with a MARR of 9%.


Benefit-Cost Ratio Analysis
Analysis Specifications
Dates
Constant MARR:
MARR List
Create Benefit-Cost Ratio Summary
Benefit-Cost Ratio Summary
Save Data As...
Return
Help

Constant MARR: 9.00

Date
01JAN2000

Date	MARR

Date	nvst1	nvst2	nvst3	nvst4	nvst5
01JAN2000	0.9724	1.0000	0.9724	1.1349	1.0807

Figure 46.5. Performing a Benefit-Cost Ratio Analysis

Computing a Uniform Periodic Equivalent

Suppose you need a warehouse for ten years. You have two options:

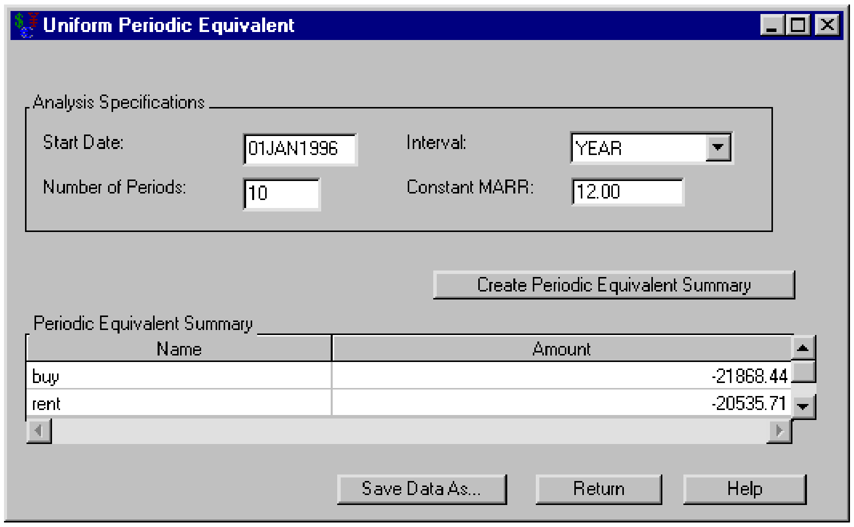
- Pay rent for ten years at $23,000 per year.
- Build a two-stage facility that you will maintain and which you intend to sell at the end of those ten years.

Datasets describing these scenarios are available in the portfolio SASHELP.INVSAMP.BUYRENT. Which option is more financially sound if you desire a 12% MARR?

Open the portfolio SASHELP.INVSAMP.BUYRENT and compare the options.

To perform the periodic equivalent, follow these steps:

1. Load the portfolio SASHELP.INVSAMP.BUYRENT.
2. Select both cashflows.
3. Select **Analyze** → **Periodic Equivalent...**.
   This opens the Uniform Periodic Equivalent dialog box.
4. Enter 01JAN1996 for the **Start Date**.
5. Enter 10 for the **Number of Periods**.
6. Select YEAR for the **Interval**.
7. Enter 12 for the **Constant MARR**.
8. Click **Create Time Value Summary**.


Uniform Periodic Equivalent
Create Periodic Equivalent Summary
Save Data As...
Return
Help

Analysis Specifications

Start Date:	01JAN1996	Interval:	YEAR
Number of Periods:	10	Constant MARR:	12.00

Periodic Equivalent Summary

Name	Amount
buy	-21868.44
rent	-20535.71

Figure 46.6. Computing a Uniform Periodic Equivalent

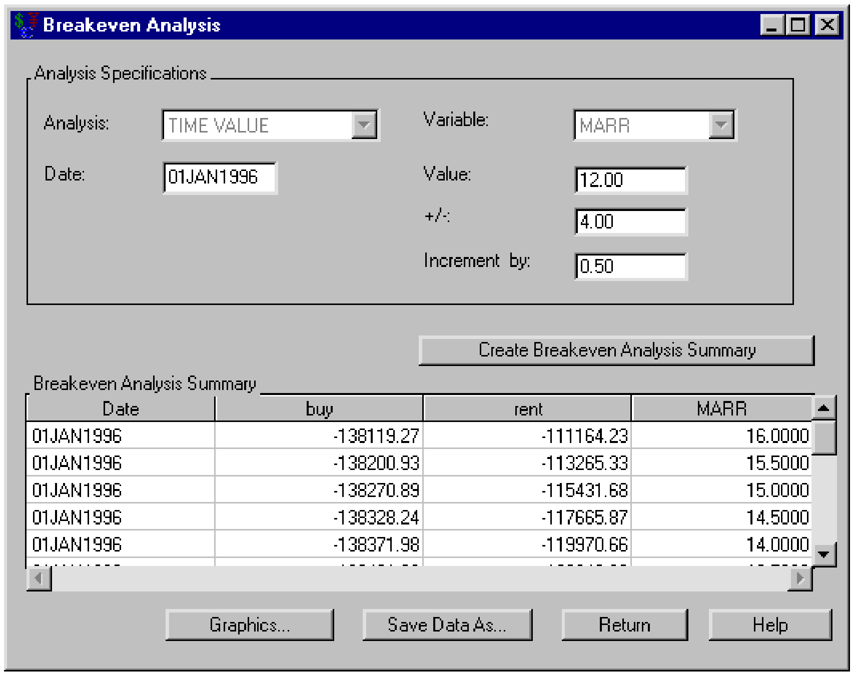
Figure 46.6 indicates that to rent costs about $1,300 less each year. Hence, renting is more financially sound. Notice the periodic equivalent for renting is not $23,000. This is because the the $23,000 per year does not account for the MARR.

Performing a Breakeven Analysis

In the previous example you computed the uniform periodic equivalent for a rent-buy scenario. Now let's perform a breakeven analysis to see how the MARR affects the time values.

To perform the breakeven analysis, follow these steps:

1. Select both options.
2. Select **Analyze** → **Breakeven Analysis...**.
3. Enter 01JAN1996 for the **Date**.
4. Enter 12.0 for **Value**.
5. Enter 4.0 for **(+/-)**.
6. Enter 0.5 for **Increment by**.
7. Click **Create Breakeven Analysis Summary** to fill the **Breakeven Analysis Summary** area as displayed in Figure 46.7.



Analysis Specifications

Analysis:	TIME VALUE	Variable:	MARR
Date:	01JAN1996	Value:	12.00
		+/-:	4.00
		Increment by:	0.50

Create Breakeven Analysis Summary

Breakeven Analysis Summary

Date	buy	rent	MARR
01JAN1996	-138119.27	-111164.23	16.0000
01JAN1996	-138200.93	-113265.33	15.5000
01JAN1996	-138270.89	-115431.68	15.0000
01JAN1996	-138328.24	-117665.87	14.5000
01JAN1996	-138371.98	-119970.66	14.0000

Figure 46.7. Performing a Breakeven Analysis

Click **Graphics...** to view a plot displaying the relationship between time value and MARR.

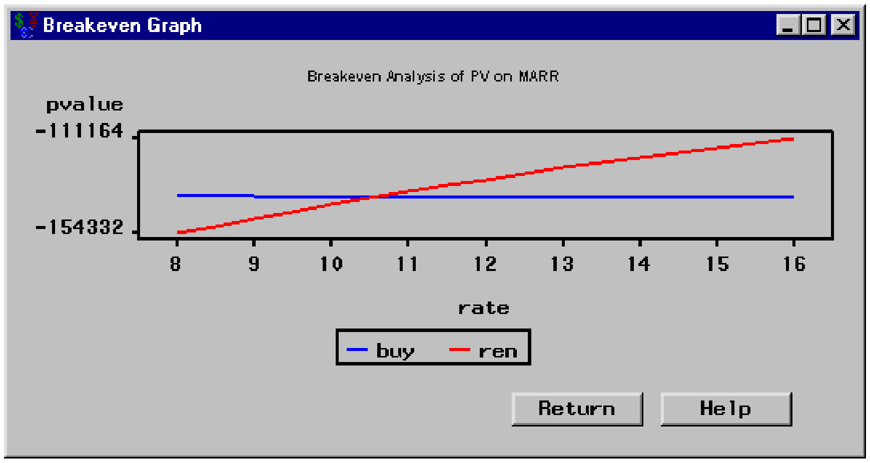


Figure 46.8. Viewing a Breakeven Graph

As shown in Figure 46.8 renting is better if you want a MARR of 12%. However, if your MARR should drop to 10.5%, buying would be better.

With a single investment, knowing where the graph has a time value of 0 tells the MARR when a venture switches from being profitable to a loss. With multiple investments, knowing where the graphs for the various investments cross each other tells at what MARR a particular investment becomes more profitable than another.

Dialog Box Guide

Time Value Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an time value analysis, select **Analyze** → **Time Value...** from the Investment Analysis dialog box's menu bar. This opens the Time Value Analysis dialog box displayed in Figure 46.9.

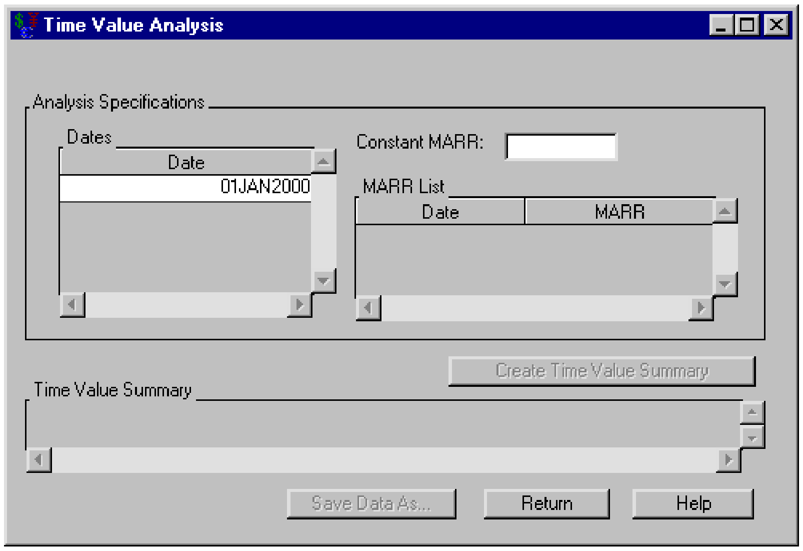


Figure 46.9. Time Value Analysis Dialog Box

The following items are displayed:

Analysis Specifications

Dates holds the list of dates as of which to perform the time value analysis. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

Constant MARR holds the desired MARR for the time value analysis. This value is used if the **MARR List** area is empty.

MARR List holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

Create Time Value Summary becomes available when you adequately specify the analysis within the **Analysis Specifications** area. Clicking **Create Time Value Summary** then fills the **Time Value Summary** area.

Time Value Summary fills when you click **Create Time Value Summary**. The table contains a row for each date in the **Dates** area. The remainder of each row holds the time values at that date, one value for each investment selected.

Print becomes available when you fill the time value summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you fill the time value summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Uniform Periodic Equivalent

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a uniform periodic equivalent, select **Analyze** → **Periodic Equivalent...** from the Investment Analysis dialog box's menu bar. This opens the Uniform Periodic Equivalent dialog box displayed in Figure 46.10.

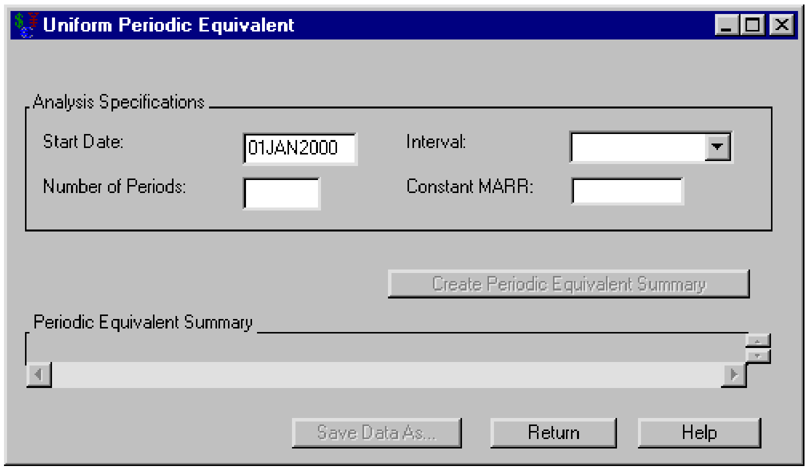


Figure 46.10. Uniform Periodic Equivalent Dialog Box

The following items are displayed:

Analysis Specifications

Start Date holds the date the uniform periodic equivalents begin.

Number of Periods holds the number of uniform periodic equivalents.

Interval holds how often the uniform periodic equivalents occur.

Constant MARR holds the Minimum Attractive Rate of Return.

Create Periodic Equivalent Summary becomes available when you adequately fill the **Analysis Specification** area. Clicking **Create Periodic Equivalent Summary** then fills the periodic equivalent summary.

Periodic Equivalent Summary fills with two columns when you click **Create Periodic Equivalent Summary**. The first column lists the investments selected. The second column lists the computed periodic equivalent amount.

Print becomes available when you fill the periodic equivalent summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you generate the periodic equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Internal Rate of Return

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an internal rate of return calculation, select **Analyze** $\rightarrow$ **Internal Rate of Return...** from the Investment Analysis dialog box's menu bar. This opens the Internal Rate of Return dialog box displayed in Figure 46.11.

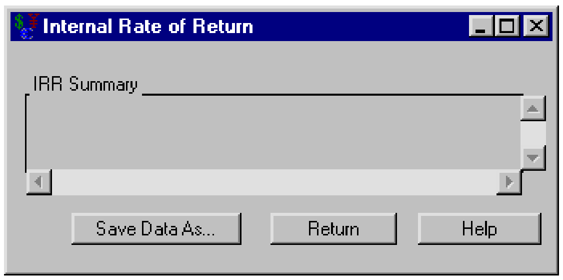


Figure 46.11. Internal Rate of Return Dialog Box

The following items are displayed:

IRR Summary contains a row for each deposit. Each row holds:

> **Name** holds the name of the investment.
>
> **IRR** holds the internal rate of return for that investment.
>
> **interval** holds the interest rate interval for that **IRR**.

Print becomes available when you fill the IRR summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... Clicking it opens the Save Output Dataset dialog box where you can save the IRR summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Benefit-Cost Ratio Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to compute a benefit-cost ratio, select **Analyze** → **Benefit-Cost Ratio...** from the Investment Analysis dialog box's menu bar. This opens the Benefit-Cost Ratio Analysis dialog box displayed in Figure 46.12.

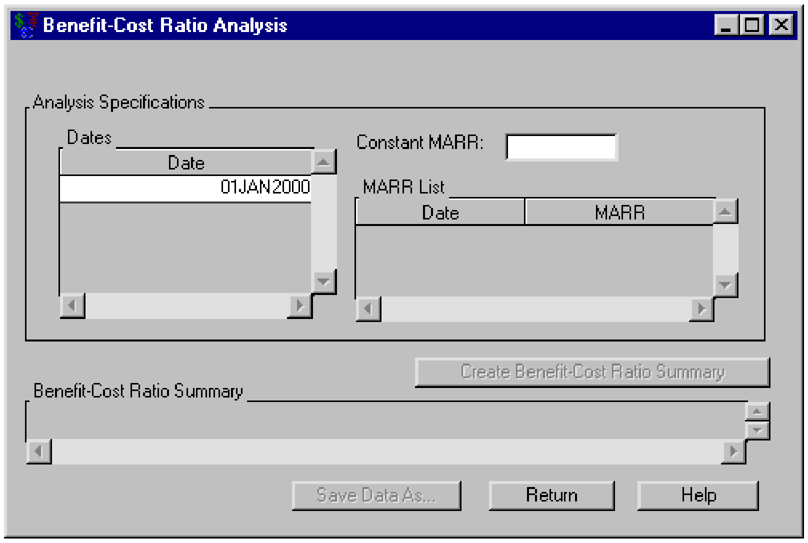


Figure 46.12. Benefit-Cost Ratio Analysis Dialog Box

The following items are displayed:

Analysis Specifications

> **Dates** holds the dates as of which to compute the Benefit-Cost ratios.
>
> **Constant MARR** holds the desired MARR. This value is used if the **MARR List** area is empty.
>
> **MARR List** holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

Create Benefit-Cost Ratio Summary becomes available when you adequately specify the analysis. Clicking **Create Benefit-Cost Ratio Summary** fills the benefit-cost ratio summary.

Benefit-Cost Ratio Summary fills when you click **Exchange the Rates**. The area contains a row for each date in the **Dates** area. The remainder of each row holds the benefit-cost ratios at that date, one value for each investment selected.

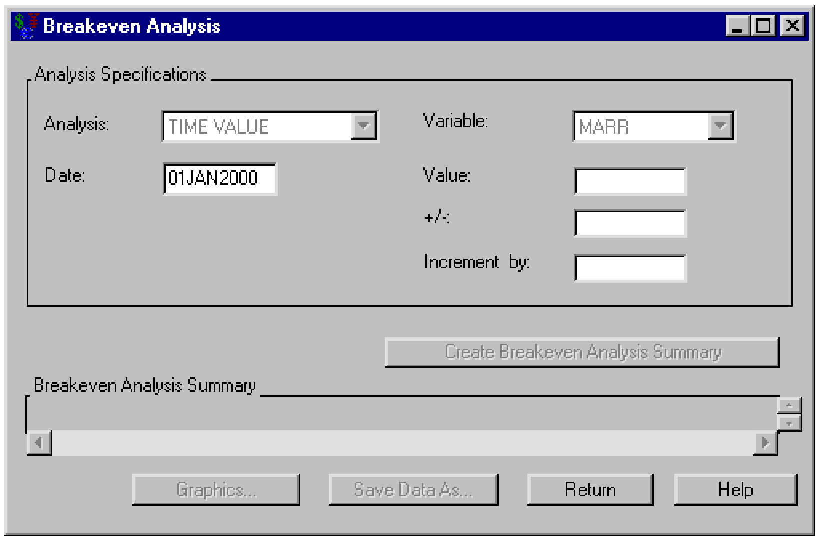
Print becomes available when you fill the benefit-cost ratio summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you generate the benefit-cost ratio summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Breakeven Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a breakeven analysis, select **Analyze** → **Breakeven Analysis...** from the Investment Analysis dialog box's menu bar. This opens the Breakeven Analysis dialog box displayed in Figure 46.13.



Analysis: TIME VALUE
Date: 01JAN2000
Variable: MARR
Value:
+/-:
Increment by:

Figure 46.13. Breakeven Analysis Dialog Box

The following items are displayed:

Analysis Specification

Analysis holds the analysis type. Only Time Value is currently available.

Date holds the date for which you perform this analysis.

Variable holds the variable upon which the breakeven analysis will vary. Only MARR is currently available.

Value holds the desired rate upon which to center the analysis.

+/- holds the maximum deviation from the **Value** to consider.

Increment by holds the increment by which the analysis is calculated.

Create Breakeven Analysis Summary becomes available when you adequately specify the analysis. Clicking **Create Breakeven Analysis Summary** then fills the **Breakeven Analysis Summary** area.

Breakeven Analysis Summary fills when you click **Create Breakeven Analysis Summary**. The schedule contains a row for each MARR and date.

Graphics... becomes available when you fill the **Breakeven Analysis Summary** area. Clicking it opens the Breakeven Graph graph representing the time value versus MARR.

Print becomes available when you fill the breakeven analysis summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As... becomes available when you generate the breakeven analysis summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Breakeven Graph

Suppose you perform a breakeven analysis in the Breakeven Analysis dialog box. Once you create the breakeven analysis summary, you can click the **Graphics...** button to open the Breakeven Graph dialog box displayed in Figure 46.14.

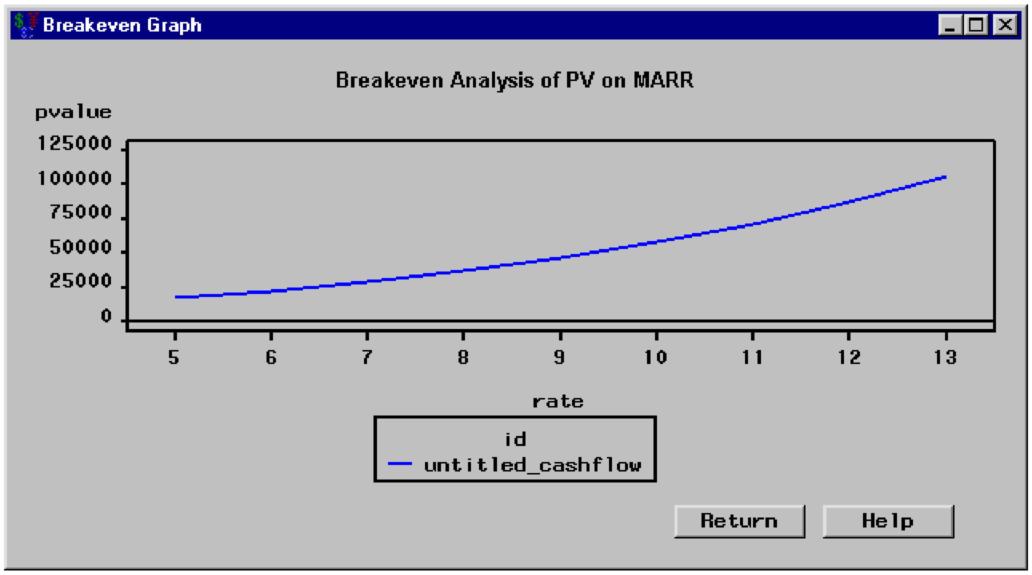


Figure 46.14. Breakeven Graph Dialog Box

The following item is displayed:

Return takes you back to the Breakeven Analysis dialog box.

Chapter 47
Details

Chapter Contents

Chapter 47
Details

Investments and Datasets

Investment Analysis provides tools to assist you in moving data between SAS data sets and lists you can use within Investment Analysis.

Saving Output to SAS Datasets

Many investment specifications have a button that reads **Save Data As...**. Clicking that button opens the Save Output Dataset dialog box (see Figure 47.1). This dialog box enables you to save all or part of the area generated by the specification.

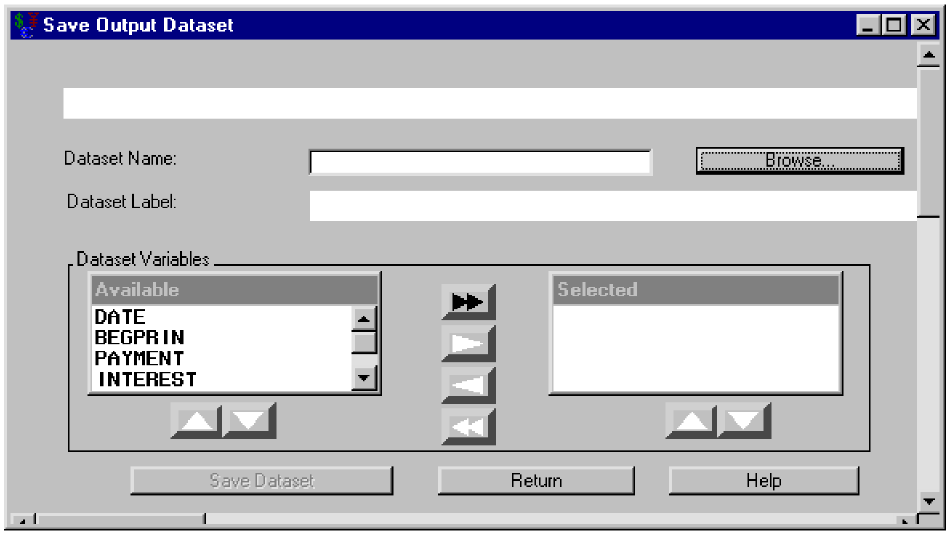


Figure 47.1. Saving to a Dataset

The following items are displayed:

Dataset Name holds the SAS data set name to which you wish to save.

Browse... opens the standard SAS **Open** dialog box, which enables you to select an existing SAS data set to overwrite.

Dataset Label holds the SAS data set's label.

Dataset Variables organizes variables. The variables listed in the **Selected** area will be included in the SAS data set.

- You can select variables one at a time, by clicking the single right-arrow after each selection to move it to the **Selected** area.

- If the desired SAS data set has many variables you wish to save, it may be simpler to follow these steps:
    1. Click the double right arrow to select all available variables.
    2. Remove any unwanted variable by selecting it from the **Selected** area and clicking the single left arrow.
- The double left arrow removes all selected variables from the proposed SAS data set.
- The up and down arrows below the **Available** and **Selected** boxes enable you to scroll up and down the list of variables in their respective boxes.

Save Dataset attempts to save the SAS data set. If the SAS data set name exists, you are asked if you want to replace the existing SAS data set, append to the existing SAS data set, or cancel the current save attempt. You then return to this dialog box ready to create another SAS data set to save.

Return takes you back to the specification dialog box.

Loading a SAS Dataset into a List

Right-click in the area you wish to load the list and release on **Load...**. This opens the Load Dataset dialog box (see Figure 47.2).

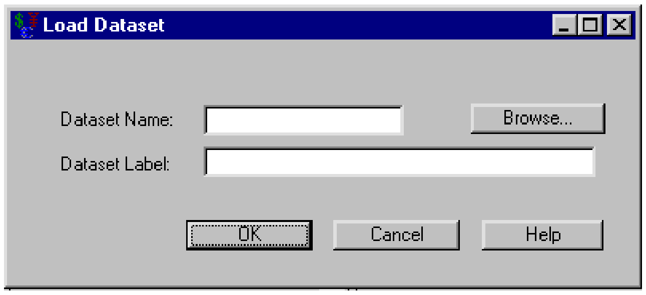


Figure 47.2. Load Dataset Dialog Box

The following items are displayed:

Dataset Name holds the name of the SAS data set you wish to load.

Browse... opens the standard SAS **Open** dialog box, which aids in finding a SAS data set to load. If there is a **Date** variable in the SAS data set, Investment Analysis loads it into the list. If there is no **Date** variable, it loads the first available time-formatted variable. If an amount or rate variable is needed, Investment Analysis searches the SAS data set for a **Amount** or **Rate** variable to use. Otherwise it takes the first numeric variable that is not used by the **Date** variable.

Dataset Label holds a SAS data set label.

OK attempts to load the SAS data set specified in **Dataset Name**. If the specified SAS data set exists, clicking **OK** returns you to the calling dialog box with the selected

SAS data set filling the list. If the specified SAS data set does not exist and you click **OK**, you receive an error message and no SAS data set is loaded.

Cancel returns you to the calling dialog box without loading a SAS data set. To load values from a SAS data set into a list, follow these steps:

Saving Data from a List to a SAS Dataset

Right-click in the area you wish to hold the list, and release on **Save...**. This opens the Save Dataset dialog box.

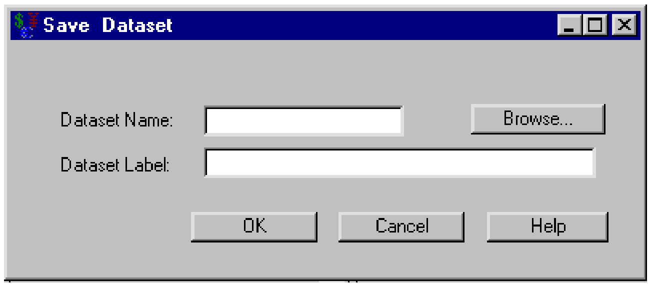


Figure 47.3. Save Dataset Dialog Box

The following items are displayed:

Dataset Name holds the SAS data set name to which you wish to save.

Browse... opens the standard SAS **Save As** dialog box, which enables you to find an existing SAS data set to overwrite.

Dataset Label holds a user-defined description to be saved as the label of the SAS data set.

OK saves the current data to the SAS data set specified in **Data set Name**. If the specified SAS data set does not already exist, clicking **OK** saves the SAS data set and returns you to the calling dialog box. If the specified SAS data set does already exist, clicking **OK** warns you and enables you to replace the old SAS data set with the new SAS data set or cancel the save attempt.

Cancel aborts the save process. Clicking **Cancel** returns you to the calling dialog box without attempting to save.

Right Mouse Button Options

A pop-up menu often appears when you right-click within table editors. The menus offer tools to aid in the management of the table's entries. Most table editors provide the following options.

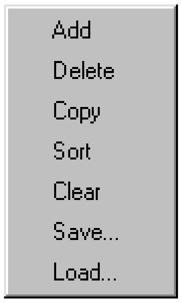


Figure 47.4. Right-Clicking Options

Add creates a blank row.

Delete removes any currently selected row.

Copy duplicates the currently selected row.

Sort arranges the rows in chronological order according to the date variable.

Clear empties the table of all rows.

Save... opens the Save Dataset dialog box where you can save the all rows to a SAS Dataset for later use.

Load... opens the Load Dataset dialog box where you select a SAS Dataset to fill the rows.

If you wish to perform one of these actions on a collection of rows, you must select a collection of rows before right-clicking. To select an adjacent list of rows, do the following: click the first pair, hold down SHIFT, and click the final pair. Once the list of rows is selected, you may release the SHIFT key.

Depreciation Methods

Suppose an asset's price is $20,000 and it has a salvage value of $5,000 in five years. The following sections describe various methods to quantify the depreciation.

Straight Line (SL)

This method assumes a constant depreciation value per year.

Assuming the price of a depreciating asset is P and its salvage value after N years is S,

$$\text{Annual Depreciation} = \frac{P-S}{N}$$

For our example, the annual depreciation would be

$$\frac{\$20,000 - \$5,000}{5} = \$3,000$$

Sum-of-years Digits

An asset often loses more of its value early in its lifetime. A method that exhibits this dynamic is desirable.

Assume an asset depreciates from price P to salvage value S in N years. First compute the value: sum-of-years = $1 + 2 + \cdots + N$. The depreciation for the years after the asset's purchase is:

Table 47.1. Sum-of-years General Example

year number	annual depreciation
first	$\frac{N}{\mathrm{sum-of-years}}(P-S)$
second	$\frac{N-1}{\mathrm{sum-of-years}}(P-S)$
third	$\frac{N-2}{\mathrm{sum-of-years}}(P-S)$
$\vdots$	$\vdots$
final	$\frac{1}{\mathrm{sum-of-years}}(P-S)$

For the ith year of the asset's use this equation generalizes to

$$\text{Annual Depreciation} = \frac{N+1-i}{\mathrm{sum-of-years}}(P-S)$$

For our example, $N = 5$ and the sum of years is $1+2+3+4+5 = 15$. The depreciation during the first year is

$$(\$20,000 - \$5,000)\frac{5}{15} = \$5,000$$

Table 47.2 describes how Declining Balance would depreciate the asset.

Table 47.2. Sum-of-years Example

Year	Depreciation	Year-end Value
1	$(\$20,000 - \$5,000)\frac{5}{15} = \$5,000$	$\$15,000.00$
2	$(\$20,000 - \$5,000)\frac{4}{15} = \$4,000$	$\$11,000.00$
3	$(\$20,000 - \$5,000)\frac{3}{15} = \$3,000$	$\$8,000.00$
4	$(\$20,000 - \$5,000)\frac{2}{15} = \$2,000$	$\$6,000.00$
5	$(\$20,000 - \$5,000)\frac{1}{15} = \$1,000$	$\$5,000.00$

And as expected, the value after N years is S.

$$\begin{aligned}
\text{Value after 5 years} &= P \text{ - (5 years' depreciation)} \\
&= P \text{ - } \Big(\tfrac{5}{10}(P-S) + \tfrac{4}{10}(P-S) + \tfrac{3}{10}(P-S) + \\
&\qquad \tfrac{2}{10}(P-S) + \tfrac{1}{10}(P-S)\Big) \\
&= P - (P-S) \\
&= S
\end{aligned}$$

Declining Balance (DB)

Recall that the Straight Line method assumes a constant depreciation value. Conversely, the Declining Balance method assumes a constant depreciation rate per year. And like the Sum-of-years method, more depreciation tends to occur earlier in the asset's life.

Assume the price of a depreciating asset is P and its salvage value after N years is S. You could assume the asset depreciates by a factor of $\frac{1}{N}$ (or a rate o f $\frac{100}{N}\%$). This method is known as Single Declining Balance. In an equation this looks like:

$$\text{Annual Depreciation} = \tfrac{1}{N}\ \text{Previous year's value}$$

So for our example, the depreciation during the first year is

$$\frac{\$20,000}{5} = \$4,000$$

Table 47.3 describes how Declining Balance would depreciate the asset.

Table 47.3. Declining Balance Example

Year	Depreciation	Year-end Value
1	$\frac{\$20{,}000.00}{5} = \$4,000.00$	$\$16,000.00$
2	$\frac{\$16{,}000.00}{5} = \$3,200.00$	$\$12,800.00$
3	$\frac{\$12{,}800.00}{5} = \$2,560.00$	$\$10,240.00$
4	$\frac{\$10{,}240.00}{5} = \$2,560.00$	$\$8,192.00$
5	$\frac{\$12{,}800.00}{5} = \$2,560.00$	$\$6,553.60$

DB Factor

You could also accelerate the depreciation by increasing the factor (and hence the rate) at which depreciation occurs. Other commonly accepted depreciation rates are $\frac{200}{N}\%$ (called Double Declining Balance as the depreciation factor becomes $\frac{2}{N}$) and $\frac{150}{N}\%$. Investment Analysis enables you to choose between these three types for Declining Balance: 2 (with $\frac{200}{N}\%$ depreciation), 1.5 (with $\frac{150}{N}\%$), and 1 (with $\frac{100}{N}\%$).

Declining Balance and the Salvage Value

The Declining Balance method assumes that depreciation is faster earlier in an asset's life; this is what you wanted. But notice the final value is greater than the salvage value. Even if the salvage value were greater than $6,553.60, the final year-end value would not change. The salvage value never enters the calculation, so there is no way for the salvage value to force the depreciation to assume its value. Newnan and Lavelle (1998) describe two ways to adapt the Declining Balance method to assume the salvage value at the final time. One way is as follows:

Suppose you call the depreciated value after i years $V(i)$. This sets $V(0) = P$ and $V(N) = S$.

- If $V(N) > S$ according to the usual calculation for $V(N)$, redefine $V(N)$ to equal S.
- If $V(i) < S$ according to the usual calculation for $V(i)$ for some i (and hence for all subsequent $V(i)$ values), you can redefine all such $V(i)$ to equal S.

This alteration to Declining Balance forces the depreciated value of the asset after N years to be S and keeps $V(i)$ no less than S.

Conversion to SL

The second (and preferred) way to force Declining Balance to assume the salvage value is by Conversion to Straight Line. If $V(N) > S$, the first way redefines $V(N)$ to equal S; you can think of this as converting to the Straight Line method for the last timestep.

If the $V(N)$ value supplied by DB is appreciably larger than S, then the depreciation in the final year would be unrealistically large. An alternate way is to compute the DB and SL step at each timestep and take whichever step gives a larger depreciation (unless DB drops below the salvage value).

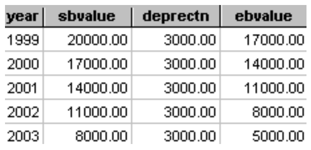
Once SL assumes a larger depreciation, it continues to be larger over the life of the asset. This forces the value at the final time to equal the salvage value as SL forces this. As an algorithm, this looks like

```
V(0) = P;
for i=1 to N
   if DB step > SL step from (i,V(i))
      take a DB step to make V(i);
   else
      break;
for j = i to N
   take a SL step to make V(j);
```

The MACRS discussed in Depreciation Table... is actually a variation on the Declining Balance with conversion to Straight Line method.

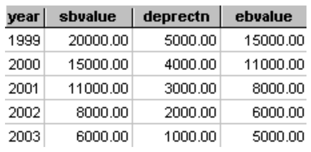
Comparison of Depreciation Methods

Figure 47.5 through Figure 47.8 display the depreciation for four depreciation methods. This example also assumes the asset has an initial value of $20,000 and depreciates to $5,000 in five years.



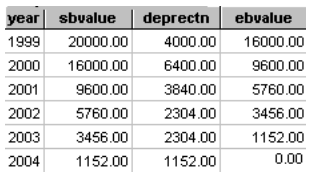
year	sbvalue	deprectn	ebvalue
1999	20000.00	3000.00	17000.00
2000	17000.00	3000.00	14000.00
2001	14000.00	3000.00	11000.00
2002	11000.00	3000.00	8000.00
2003	8000.00	3000.00	5000.00

Figure 47.5. Straight Line



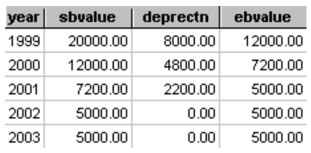
year	sbvalue	deprectn	ebvalue
1999	20000.00	5000.00	15000.00
2000	15000.00	4000.00	11000.00
2001	11000.00	3000.00	8000.00
2002	8000.00	2000.00	6000.00
2003	6000.00	1000.00	5000.00

Figure 47.6. Sum-of-years Digits



year	sbvalue	deprectn	ebvalue
1999	20000.00	4000.00	16000.00
2000	16000.00	6400.00	9600.00
2001	9600.00	3840.00	5760.00
2002	5760.00	2304.00	3456.00
2003	3456.00	2304.00	1152.00
2004	1152.00	1152.00	0.00

Figure 47.7. Depreciation Table



year	sbvalue	deprectn	ebvalue
1999	20000.00	8000.00	12000.00
2000	12000.00	4800.00	7200.00
2001	7200.00	2200.00	5000.00
2002	5000.00	0.00	5000.00
2003	5000.00	0.00	5000.00

Figure 47.8. Declining Balance

- Under Depreciation Table, realize a 5-year class MACRS Depreciation actually lasts 6 years.
- The Declining Balance method is Double Declining Balance with conversion to Straight Line.

For further reference, consider Newnan and Lavelle (1998). They offer explanations and graphs for the individual depreciation methods. They also analyze the differences between the various methods.

Rate Information

The Tools Menu

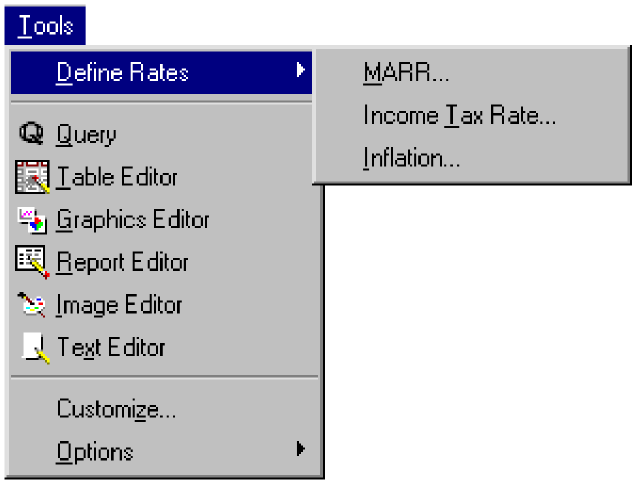


Figure 47.9. The Tools Menu

The **Tools** → **Define Rates** menu offers the following options.

MARR... opens the Minumum Attractive Rate of Return (MARR) dialog box.

Income Tax Rate... opens the Income Tax Specification dialog box.

Inflation... opens the Inflation Specification dialog box.

Dialog Box Guide

Minimum Attractive Rate of Return (MARR)

Selecting **Tools** → **Define Rates** → **MARR** from the Investment Analysis dialog box menu bar opens the MARR dialog box displayed in Figure 47.10.

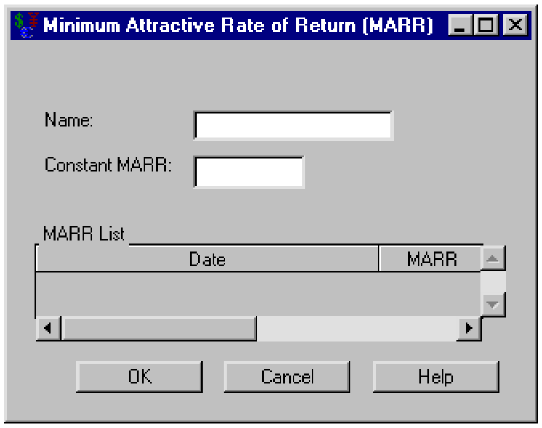


Figure 47.10. MARR Dialog Box

Name holds the name you assign to the MARR specification. This must be a valid SAS name.

Constant MARR holds the numeric value you choose to be the constant MARR. This value is used if the **MARR List** table editor is empty.

MARR List holds date-MARR pairs where the date refers to when the particular MARR value begins. Each date is a SAS date.

OK returns you to the Investment Analysis dialog box. Pressing it causes the preceding MARR specification to be assumed when you do not specify MARR rates in a dialog box that needs MARR rates.

Cancel returns you to the Investment Analysis dialog box, discarding any work done in the MARR dialog box.

Income Tax Specification

Selecting **Tools** → **Define Rates** → **Income Tax Rate** from the Investment Analysis dialog box menu bar opens the Income Tax Specification dialog box displayed in Figure 47.11.

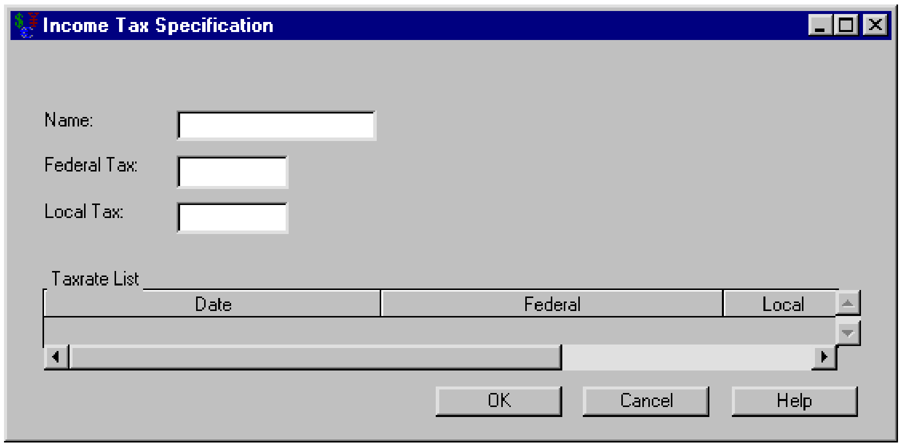


Figure 47.11. Income Tax Specification Dialog Box

Name holds the name you assign to the Income Tax specification. This must be a valid SAS name.

Federal Tax holds the numeric value you desire to be the constant Federal Tax.

Local Tax holds the numeric value you desire to be the constant Local Tax.

Taxrate List holds date-Income Tax triples where the date refers to when the particular Income Tax value begins. Each date is a SAS date, and the value is a percentage between 0% and 100%.

OK returns you to the Investment Analysis dialog box. Clicking it causes the preceding income tax specification to be the default income tax rates when using the After Tax Cashflow Calculation dialog box.

Cancel returns you to the Investment Analysis dialog box, discarding any editions since this dialog box was opened.

Inflation Specification

Selecting **Tools** → **Define Rates** → **Inflation** from the Investment Analysis dialog box menu bar opens the Inflation Specification dialog box displayed in Figure 47.12.

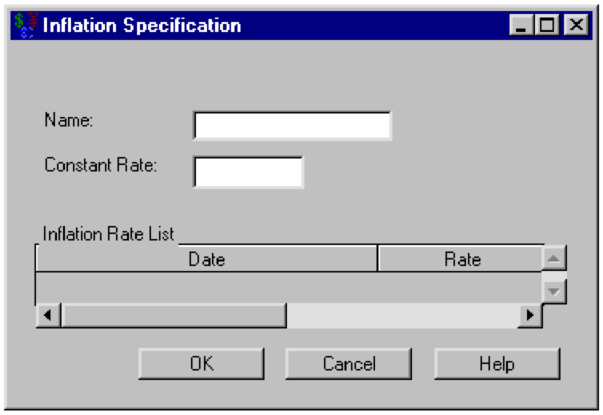


Figure 47.12. Inflation Specification Dialog Box

Name holds the name you assign to the Inflation specification. This must be a valid SAS name.

Constant Rate holds the numeric value you desire to be the constant inflation rate. This value is used if the **Inflation Rate List** table editor is empty.

Inflation Rate List holds date-rate pairs where the date refers to when the particular inflation rate begins. Each date is a SAS date and the rate is a percentage between 0% and 120%.

OK returns you to the Investment Analysis dialog box. Pressing it causes the preceding inflation specification to be assumed when you use the Constant Dollar Calculation dialog box and do not specify inflation rates.

Cancel returns you to the Investment Analysis dialog box, discarding any editions since this dialog box was opened.

Reference

Newnan, Donald G. and Lavelle, Jerome P. (1998), *Engineering Economic Analysis*, Austin, Texas: Engineering Press.

Subject Index

Syntax Index

Your Turn

If you have comments or suggestions about *SAS/ETS® 9.1 User's Guide,* please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **`yourturn@sas.com`**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **`suggest@sas.com`**