SAS Publishing

SAS® High-Performance Forecasting 2.2
User's Guide

THE POWER TO KNOW®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *SAS® High-Performance Forecasting 2.2: User's Guide, Volumes 1 and 2*. Cary, NC: SAS Institute Inc.

SAS® High-Performance Forecasting 2.2: User's Guide, Volumes 1 and 2

ISBN-13: 978-1-59994-168-4
ISBN-10: 1-59994-168-6

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

1st printing, August 2006

Contents

Acknowledgments

Credits

Documentation

Editing	Ed Huddleston
Technical Review	Bruce Elsheimer, Wilma S. Jackson, Tae Yoon Lee, Michael J. Leonard, Kevin Meyer, Rajesh Selukar, Donna E. Woodward
Documentation Production	Tim Arnold

Software

The procedures and functions in SAS High-Performance Forecasting 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 or function are given first.

HPF Procedure	Michael J. Leonard
HPFARIMASPEC	Rajesh Selukar, Keith Crowe
HPFDIAGNOSE	Youngjin Park, Michael J. Leonard, Rajesh Selukar
HPFENGINE	Keith Crowe, Michael J. Leonard, Rajesh Selukar
HPFESMSPEC	Michael J. Leonard, Keith Crowe
HPFEVENTS	Wilma S. Jackson, Michael J. Leonard
HPFEXMSPEC	Michael J. Leonard, Keith Crowe
HPFIDMSPEC	Michael J. Leonard, Keith Crowe
HPFRECONCILE	Mahesh Joshi, Michele Trovero
HPFSELECT	Keith Crowe, Michael J. Leonard

HPFUCMSPEC	Rajesh Selukar, Keith Crowe
HPFSCSIG function	Keith Crowe
HPFSCSUB function	Keith Crowe, Rajesh Selukar

Technical Support

Members	Donna E. Woodward, Kevin Meyer

Acknowledgments

Hundreds of people have helped improve the SAS System in many ways since its inception. The following individuals have been especially helpful in the development of the procedures in SAS High-Performance Forecasting software. Acknowledgments for the SAS System generally appear in Base SAS software documentation and HPF software documentation.

David M. DeLong	Duke University
David Dickey	North Carolina State University
Wayne Fuller	Iowa State University
Brian Monsell	U.S. Census Bureau

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 High-Performance Forecasting 2.2

Overview

Note: This section describes the features of SAS High-Performance Forecasting that are new or enhanced since SAS 9.0.

SAS High-Performance Forecasting has a new release numbering scheme. SAS High-Performance Forecasting 2.2 provides new procedures, features, and functionality while maintaining all the capabilities of SAS 9.1.3 High-Performance Forecasting software.

New features, procedures, and functions have been added to SAS High-Performance Forecasting as follows:

- New features have been added to the HPF procedure.
- New procedures have been added for creating model specifications and model selection lists:
  - PROC HPFARIMASPEC
  - PROC HPFESMSPEC
  - PROC HPFEXMSPEC
  - PROC HPFIDMSPEC
  - PROC HPFSELECT
  - PROC HPFUCMSPEC
- The new HPFEVENTS procedure has been added for creating calendar events.
- New procedures have been added for computations:
  - PROC HPFDIAGNOSE
  - PROC HPFENGINE
- The new HPFRECONCILE procedure has been added for reconciliation of hierarchical forecasts.
- New experimental functions have been added for forecast scoring:
  - HPFSCSIG function
  - HPFSCSUB function

Details

High-Performance Forecasting Release Numbering Scheme

SAS High-Performance Forecasting has a new release numbering scheme. SAS High-Performance Forecasting 2.2 provides the same features and functionality as SAS 9.1.3 High-Performance Forecasting software, and includes new procedures, functions, and features.

HPF Procedure

The HPF procedure has the following new features:

- New options related to forecast model selection:
  - HOLDOUTPCT= option
  - SEASONTEST= option
  - SELECT= option
- The new NBACKCAST= option is related to forecast model initialization.
- The new IDM statement is related to intermittent demand models.
- New options related to out-of-sample forecast performance statistics:
  - BACK= option
  - PRINT=PERFORMANCE option
  - PRINT=PERFORMANCESUMMARY option
  - PRINT=PERFORMANCEOVERALL option
- New options related to the processing of the data:
  - NOTSORTED option
  - REPLACEBACK option
  - SORTNAMES option
- New options related to printed and graphical output:
  - PLOT= option
  - PRINT=STATES option
- The new MAXERROR= option is related to the message log.

HPFARIMASPEC Procedure

The new HPFARIMASPEC procedure is used to create an Autoregressive Integrated Moving Average (ARIMA) model specification file. The output of the procedure is an XML file that stores the intended ARIMA model specification. This XML specification file can be used to populate the model repository used by the HPFENGINE procedure. (Likewise, the XML files generated by the other model specification procedures in this section can also be used to populate the model repository used by PROC HPFENGINE.)

HPFESMSPEC Procedure

The new HPFESMSPEC procedure is used to create an Exponential Smoothing Model (ESM) specification file. The output of the procedure is an XML file that stores the intended ESM model specification.

HPFEXMSPEC Procedure

The new HPFEXMSPEC procedure is used to create an External Model (EXM) specification file. The output of the procedure is an XML file that stores the intended EXM model specification.

HPFIDMSPEC Procedure

The HPFIDMSPEC procedure is used to create an Intermittent Demand Model (IDM) specification file. The output of the procedure is an XML file that stores the intended IDM model specification.

HPFSELECT Procedure

The new HPFSELECT procedure is used to create model selections lists. A model selection list contains references to candidate model specifications stored in the model repository. The output of the procedure is an XML file that stores the intended model selection list.

The HPFSELECT procedure has the following experimental features:

- external variable mapping (EXMMAP= option)
- external forecast function (EXMFUNC= option)

HPFUCMSPEC Procedure

The new HPFUCMSPEC procedure is used to create an Unobserved Component Model (UCM) specification file. The output of the procedure is an XML file that stores the intended UCM model specification.

HPFEVENTS Procedure

The HPFEVENTS procedure provides a way to create and manage events associated with time series. The procedure can create events, read events from an events data set, write events to an events data set, and create dummies based on those events, if date information is provided.

A SAS event is used to model any incident that disrupts the normal flow of the process that generated the time series. Examples of commonly used events include natural disasters, retail promotions, strikes, advertising campaigns, policy changes, and data recording errors.

An event has a reference name, a date or dates associated with the event, and a set of qualifiers. The event exists separately from any time series; however, the event may be applied to one or more time series. When the event is applied to a time series, a dummy variable is generated that may be used to analyze the impact of the event on the time series.

HPFDIAGNOSE Procedure

The new HPFDIAGNOSE procedure is an automatic modeling procedure to find the best model among ARIMA Models, Exponential Smoothing Models, and Unobserved Component Models.

The HPFDIAGNOSE procedure has the following functionality:

- intermittency test
- functional transformation test
- simple differencing and seasonal differencing test
- tentative simple ARMA order identification
- tentative seasonal ARMA order identification
- outlier detection
- significance test of events
- transfer functions identification
- intermittent demand model
- exponential smoothing model
- unobserved component model

HPFENGINE Procedure

The new HPFENGINE procedure provides large-scale automatic forecasting of transactional or time series data. The HPFENGINE procedure extends the foundation built by PROC HPF, enabling the user to determine the list of models over which automatic selection is performed.

The use of many forecast model families is supported when HPFENGINE is used in conjunction with new procedures that generate generic model specifications. Among these models are the following:

- ARIMA
- Unobserved Component Models (UCM)
- Exponential Smoothing Models (ESM)
- Intermittent Demand Models (IDM)
- External Models (EXM)

Furthermore, users can completely customize the operation by defining their own code to generate forecasts.

For models with inputs, the STOCHASTIC statement is especially helpful for automatically forecasting those inputs that have no future values.

Also supported is the generation of a portable forecast score. The output of the SCORE statement is a file or catalog entry that, when used with the new function HPFSCSUB, can be used to efficiently generate forecasts outside of the HPFENGINE procedure.

The new HPFDIAGNOSE procedure produces output that is compatible with HPFENGINE. As a result, the task of candidate model specification can be entirely automated.

HPFRECONCILE Procedure

The new HPFRECONCILE procedure provides the following functionality:

- large-scale reconciliation of hierarchical forecasts by using predictions, standard errors, and confidence limits reconciliation
- the following methods of reconciliation:
  - top-down reconciliation
  - bottom-up reconciliation
- constrained reconciliation using an interior-point quadratic programming technique that combines statistical forecasts reconciliation with judgmental constraints (overrides)
- replaces functionality of the %HPFRECON macro

HPFSCSIG Function

The experimental HPFSCSIG function generates a sample signature for subsequent use by the HPFSCSUB function.

HPFSCSUB Function

The experimental HPFSCSUB function uses score files to produce forecasts outside of the HPFENGINE procedure. Being a function, it is particularly well suited for use within other SAS programming contexts, such as the DATA step, or procedures that permit the specification of functions, such as the NLP procedure. The only input required is a reference to the score function, the horizon, and future values of any inputs.

Part 1
General Information

Contents

Chapter 1
Introduction

Chapter Contents

Chapter 1
Introduction

Overview of SAS High-Performance Forecasting Software

SAS High-Performance Forecasting software provides a large-scale automatic forecasting system. The software provides for the automatic selection of time series models for use in forecasting time-stamped data.

Given a time-stamped data set, the software provides the following automatic forecasting process:

- accumulates the time-stamped data to form a fixed-interval time series
- diagnoses the time series using time series analysis techniques
- creates a list of candidate model specifications based on the diagnostics
- fits each candidate model specification to the time series
- generates forecasts for each candidate fitted model
- selects the most appropriate model specification based on either in-sample or holdout-sample evaluation using a model selection criterion
- refits the selected model specification to the entire range of the time series
- creates a forecast score from the selected fitted model
- generate forecasts from the forecast score
- evaluates the forecast using in-sample analysis

The software also provides for out-of-sample forecast performance analysis.

For time series data without causal inputs (input variables or calendar events), the HPF procedure provides a single, relatively easy to use batch interface that supports the preceding automatic forecasting process. The HPF procedure uses exponential smoothing models (ESM) and intermittent demand models (IDM) in an automated way to extrapolate the time series. The HPF procedure is relatively simple to use and requires only one procedure call.

For time series data with or without causal inputs (input variables and/or calendar events), the software provides several procedures that provide a batch interface that supports the preceding automatic forecasting process with more complicated models. These procedures must be used in the proper sequence in order to get the desired results. Forecasting time series of this nature normally requires more than one procedure call.

Input variables are recorded in the time-stamped data set. These input variables may or may not be incorporated in time series models used to generate forecasts.

Calendar events are specified using the HPFEVENTS procedure. These event definitions are used to generate discrete-valued indicator variables or dummy variables. These event definitions are stored in a SAS data set. These indicator variables may or may not be incorporated in time series models used to generate forecasts.

Given the specified calendar events and input variables, the HPFDIAGNOSE procedure diagnoses the time series and decides which, if any, of the calendar events or input variables are determined to be useful in forecasting the time series. The HPFDIAGNOSE procedure automatically generates candidate model specifications and a model selection list using time series analysis techniques. These model specifications and model selection lists can then be used to automatically generate forecasts.

The user can specify model specifications using one of the following model specification procedures:

- PROC HPFARIMASPEC enables the user to specify one of the family of AutoRegressive Integrated Moving Average with eXogenous inputs (ARIMAX) models.
- PROC HPFESMSPEC enables the user to specify one of the family of exponential smoothing models (ESM).
- PROC HPFEXMSPEC allows the forecast to be generated by an external source.
- PROC HPFIDMSPEC enables the user to specify one of the family of intermittent demand models (IDM).
- PROC HPFSELECT enables the user to specify a model selection list. The model selection list references one or more candidate model specifications and specifies how to choose the appropriate model for a given time series.
- PROC HPFUCMSPEC enables the user to specify one of the family of unobserved component models (UCM).

Regardless of whether the model specifications or model selection lists are specified or automatically generated, the HPFENGINE procedure uses these files to automatically select an appropriate forecasting model, estimate the model parameters, and forecast the time series.

Most of the computational effort associated with automatic forecasting is time series analysis, diagnostics, model selection, and parameter estimation. Forecast scoring files summarize the time series model's parameter estimates and the final states (historical time series information). These files can be used to quickly generate the forecasts required for the iterative nature of scenario analysis, stochastic optimization, and goal seeking computations. The HPFSCSUB function can be used to score time series information.

Uses of SAS High-Performance Forecasting Software

HPF software provides tools for a wide variety of applications in business, government, and academia. Major uses of HPF procedures include: forecasting, forecast scoring, market response modeling, and time series data mining.

Contents of SAS High-Performance Forecasting Software

Procedures

HPF software includes the following SAS procedures:

HPFARIMASPEC	The HPFARIMASPEC procedure is used to create an Autoregressive Integrated Moving Average (ARIMA) model specification file. The output of the procedure is an XML file that stores the intended ARIMA model specification. This XML specification file can be used to populate the model repository used by the HPFENGINE procedure. (Likewise, the XML files generated by the other model specification procedures in this section can also be used to populate the model repository used by PROC HPFENGINE.)
HPFDIAGNOSE	The HPFDIAGNOSE procedure is an automatic modeling procedure to find the best model among ARIMA Models, Exponential Smoothing Models, and Unobserved Component Models. The HPFDIAGNOSE procedure has the following functionality: • intermittency test • functional transformation test • simple differencing and seasonal differencing test • tentative simple ARMA order identification • tentative seasonal ARMA order identification • outlier detection • significance test of events • transfer functions identification • intermittent demand model • exponential smoothing model • unobserved component model
HPFENGINE	The HPFENGINE procedure provides large-scale automatic forecasting of transactional or time series data. The HPFENGINE procedure extends the foundation built by PROC HPF, enabling the user to determine the list of models over which automatic selection is performed.

The use of many forecast model families is supported when HPFENGINE is used in conjunction with new experimental procedures that generate generic model specifications. Among these models are

- ARIMA
- Unobserved Component Models (UCM)
- Exponential Smoothing Models (ESM)
- Intermittent Demand Models (IDM)
- External Models (EXM)

Furthermore, users may completely customize the operation by defining their own code to generate forecasts.

For models with inputs, the STOCHASTIC statement is especially helpful for automatically forecasting those inputs that have no future values.

Also supported is the generation of a portable forecast score. The output of the SCORE statement is a file or catalog entry which, when used with the new function HPFSCSUB, can be used to efficiently generate forecasts outside of the HPFENGINE procedure.

The new HPFDIAGNOSE procedure produces output that is compatible with HPFENGINE. As a result, the task of candidate model specification can be entirely automated.

HPFESMSPEC

The HPFESMSPEC procedure is used to create an Exponential Smoothing Model (ESM) specification file. The output of the procedure is an XML file that stores the intended ESM model specification.

HPFEVENTS

The HPFEVENTS procedure provides a way to create and manage events associated with time series. The procedure can create events, read events from an events data set, write events to an events data set, and create dummies based on those events, if date information is provided.

A SAS event is used to model any incident that disrupts the normal flow of the process that generated the time series. Examples of commonly used events include natural disasters, retail promotions, strikes, advertising campaigns, policy changes, and data recording errors.

An event has a reference name, a date or dates associated with the event, and a set of qualifiers. The event exists separately from any time series; however, the event may be applied to one or more time series. When the event is applied to a time series, a dummy variable is generated that may be used to analyze the impact of the event on the time series.

HPFEXMSPEC	The HPFEXMSPEC procedure is used to create an External Model (EXM) specification file. The output of the procedure is an XML file that stores the intended EXM model specification.
HPFIDMSPEC	The HPFIDMSPEC procedure is used to create an Intermittent Demand Model (IDM) specification file. The output of the procedure is an XML file that stores the intended IDM model specification.
HPFRECONCILE	The HPFRECONCILE procedure reconciles forecasts of time series data at two different levels of aggregation. Optionally, the HPFRECONCILE procedure can disaggregate forecasts from upper level forecasts or aggregate forecasts from lower level forecasts. Additionally, the procedure enables the user to specify the direction and the method of reconciliation, equality constraints and bounds on the reconciled values at each point in time.
HPFSELECT	The HPFSELECT procedure is used to create model selections lists. A model selection list contains references to candidate model specifications stored in the model repository. The output of the procedure is an XML file that stores the intended model selection list.
HPFUCMSPEC	The HPFUCMSPEC procedure is used to create an Unobserved Component Model (UCM) specification file. The output of the procedure is an XML file that stores the intended UCM model specification.
HPFSCSIG	The HPFSCSIG function generates a sample signature for subsequent use by the HPFSCSUB function.
HPFSCSUB	The HPFSCSUB function uses score files to produce forecasts outside of the HPFENGINE procedure. Being a function, it is particularly well suited for use within other SAS programming contexts, such as the DATA step, or procedures that permit the specification of functions, such as the NLP procedure. The only input required is a reference to the score function, the horizon, and future values of any inputs.

About This Book

This book is a user's guide to HPF software. Since HPF 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 Organization

The new features added to HPF software since the publication of *SAS High-Performance Forecasting Software: Changes and Enhancements for Release 8.2* are summarized in “What’s New in SAS 9, 9.1, 9.1.2, and 9.1.3 SAS High-Performance Forecasting.” If you have used this software in the past, you may want to skim this chapter to see what is new.

Following the brief "What's New" section, this book is divided into three major parts.

- Part One contains general information to aid you in working with HPF software.

  The current chapter provides an overview of this software and summarizes related SAS Institute publications, products, and services.

  This chapter is followed by an "Examples" chapter.

- Part Two, the "Procedure Reference," contains the chapters that explain the SAS procedures that make up HPF software. The chapters documenting each of the procedures appear in alphabetical order by procedure name and 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 provides a summary of and computational details on the SAS High-Performance Forecasting System, an interactive forecasting menu system. Two of the chapters in Part Three document the details of the forecasting process.

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 HPF software.

Accessing the SAS High-Performance Forecasting Sample Library

The HPF Sample Library includes many examples that illustrate the use of this 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 HPF 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 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
- SAS Products (on the Contents tab)
- SAS High-Performance Forecasting

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

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 HPF software.

Related SAS Software

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

Base SAS Software

The features provided by HPF 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 HPF software. See Chapter 2 (*SAS/ETS User's Guide*) for further discussion of some of these topics as they relate to time series data and HPF 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 HPF 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 HPF 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

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.

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 HPF software because many econometric and other statistical methods not included in HPF 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 HPF 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 HPF 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

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 HPF 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 HPF 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.

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 do not find something you need in HPF software, you may find it in SAS/STAT software and in Base SAS software. If you still do not find it, look in other SAS software products or contact the SAS Institute Technical Support staff.

References

Leonard, Michael (2002), "Large-Scale Automatic Forecasting: Millions of Forecasts," Cary, North Carolina: SAS Institute, Inc.

Leonard, Michael (2004), "Large-Scale Automatic Forecasting with Inputs and Calendar Events," Cary, North Carolina: SAS Institute, Inc.

Leonard, Michael (2003), "Mining Transactional and Time Series Data," Cary, North Carolina: SAS Institute, Inc.

Leonard, Michael (2000), "Promotional Analysis and Forecasting for Demand Planning: A Practical Time Series Approach," Cary, North Carolina: SAS Institute, Inc.

Chapter 2
Integrated Example

Chapter Contents

Chapter 2
Integrated Example

Integrated Example

Example 2.1. Simple Forecasting Task

This scenario shows the use of HPF on SASHELP.PRICEDATA in the most simple forecasting task:

Forecast SALE for different regions and products (by groups) using PRICE1-PRICE17 as predictors.

The HPFDIAGNOSE procedure is used in its default mode to find good models for each series, and the HPFENGINE procedure is used to forecast using these models.

Perform the following steps.

1. Generate models using HPFDIAGNOSE OUTEST=EST.
2. Estimate model parameters for the models chosen by HPFDIAGNOSE.
3. Forecast using the estimated models (no more fitting).

```
*Step1 ;
proc hpfdiag data=sashelp.pricedata
   outest=est
   modelrepository=work.mycat
   criterion=mape;
   id date interval=month;
   by region product;
   forecast sale;
   input price1-price17;
run;
```

```
*Step2 ;
proc hpfengine data=sashelp.pricedata inest=est outest=fest
   modelrepository=work.mycat
   print=(select estimates);
   id date interval=month;
   by region product;
   forecast sale / task=select ;
   control price1-price17;
run;
```

```
*Step3 ;
proc hpfengine data=sashelp.pricedata inest=fest
   modelrepository=work.mycat
   print=(forecasts);
   id date interval=month;
   by region product;
   forecast sale / task=forecast ;
   control price1-price17;
run;
```

Chapter 3
Working with Time Series Data

Chapter Contents

Chapter 3
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 3.1.

```
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 3.1. 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/ETS User's Guide*)

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 3.1 shows some other examples of datetime constants.

Table 3.1. 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 (*SAS/ETS User's Guide*) 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 3.2.) 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 3.2 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 3.2. 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 3.2 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 3.2. 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 (*SAS/ETS User's Guide*) 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.

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 3.3. 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 28 (*SAS/ETS User's Guide*) 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 3.4 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 3.4. 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 3.3.

Table 3.3. 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 3.5 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 3.5 with the PROC FORECAST output data set shown in Figure 3.4.

```
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 3.5. 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 3.4 summarizes the basic types of intervals.

Table 3.4. 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 (*SAS/ETS User's Guide*) 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 35, "Getting Started with Time Series Forecasting," (*SAS/ETS User's Guide*) 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 3.6.

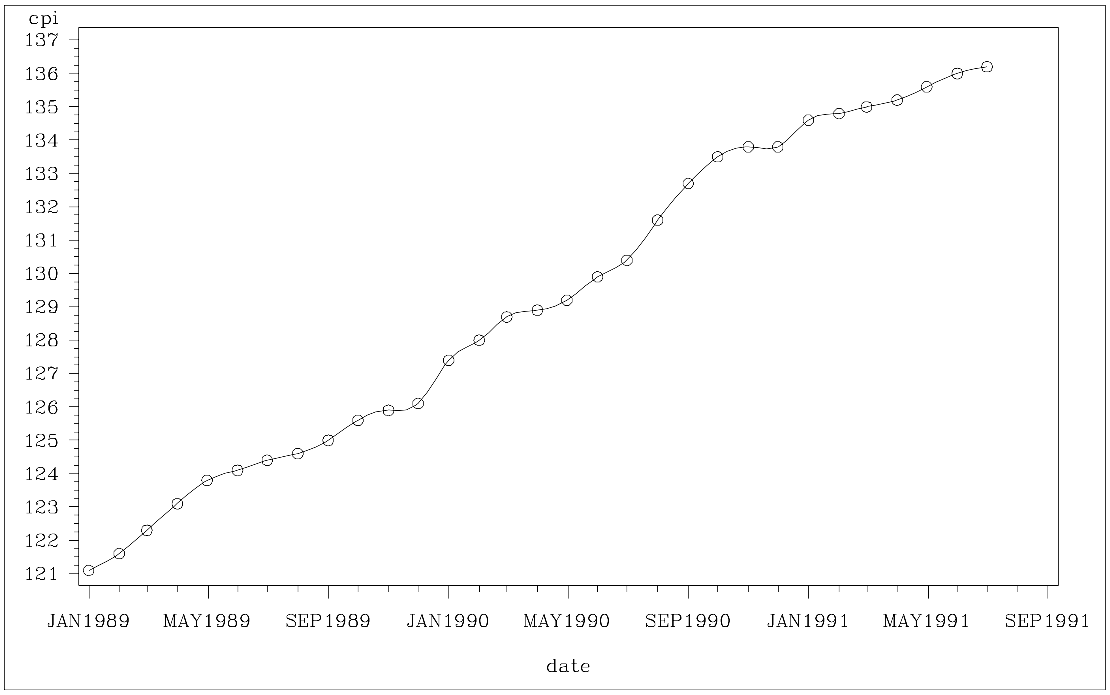


Figure 3.6. Plot of Monthly CPI Over Time

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 3.7.

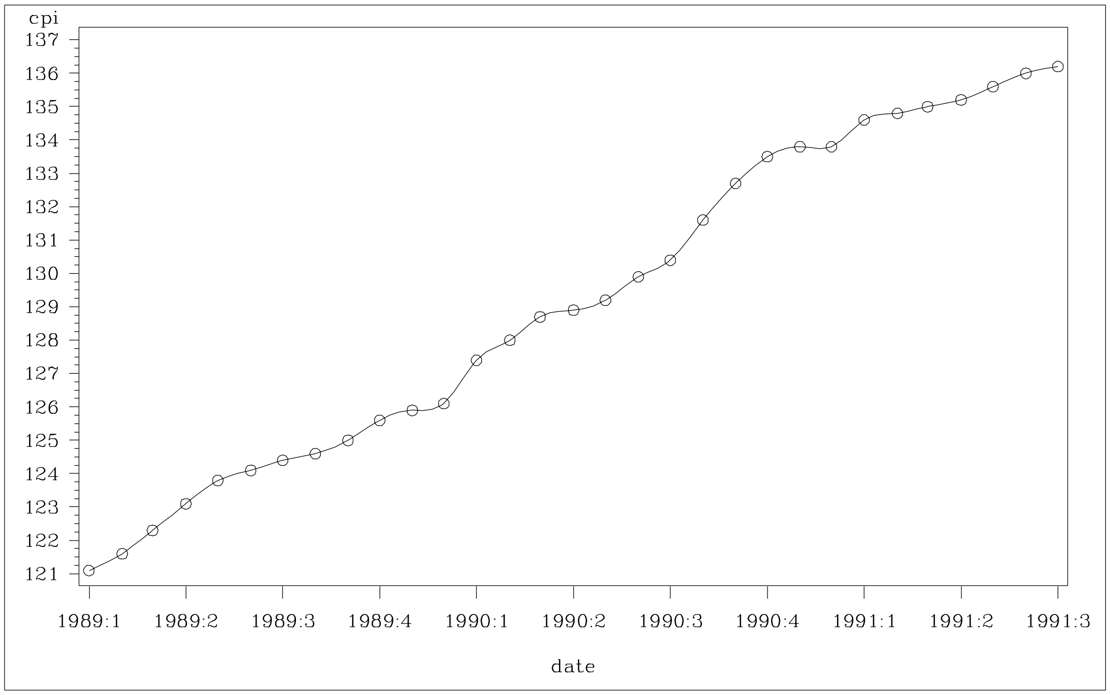


Figure 3.7. Plot of Monthly CPI Over Time

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 3.8.

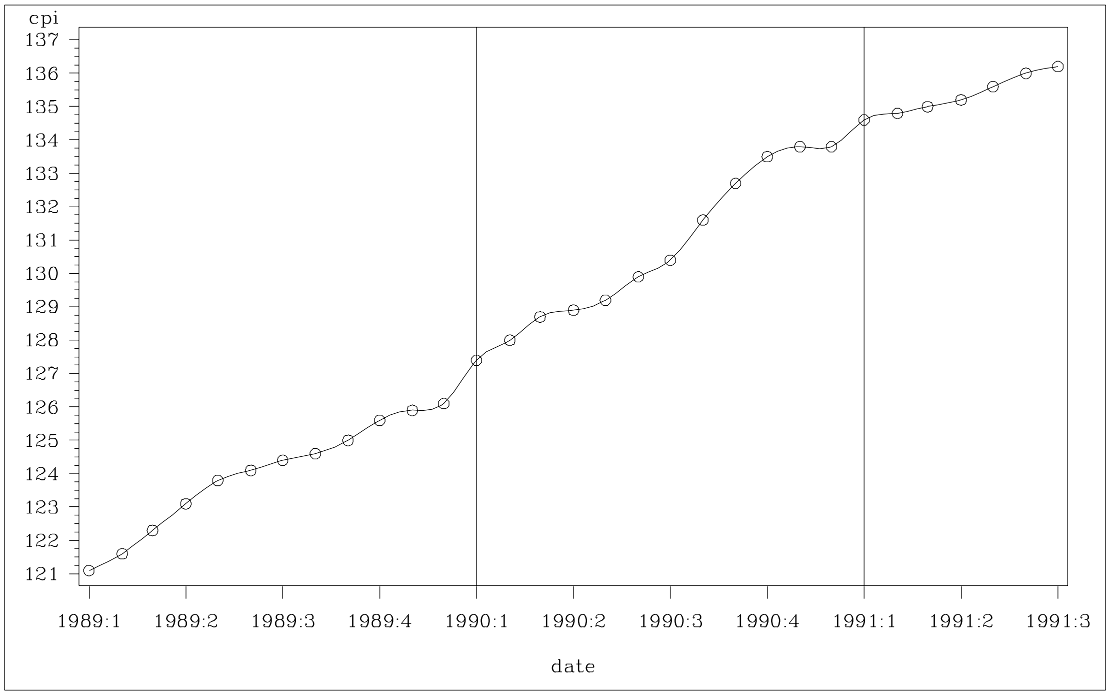


Figure 3.8. Plot of Monthly CPI Over Time

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 3.9.

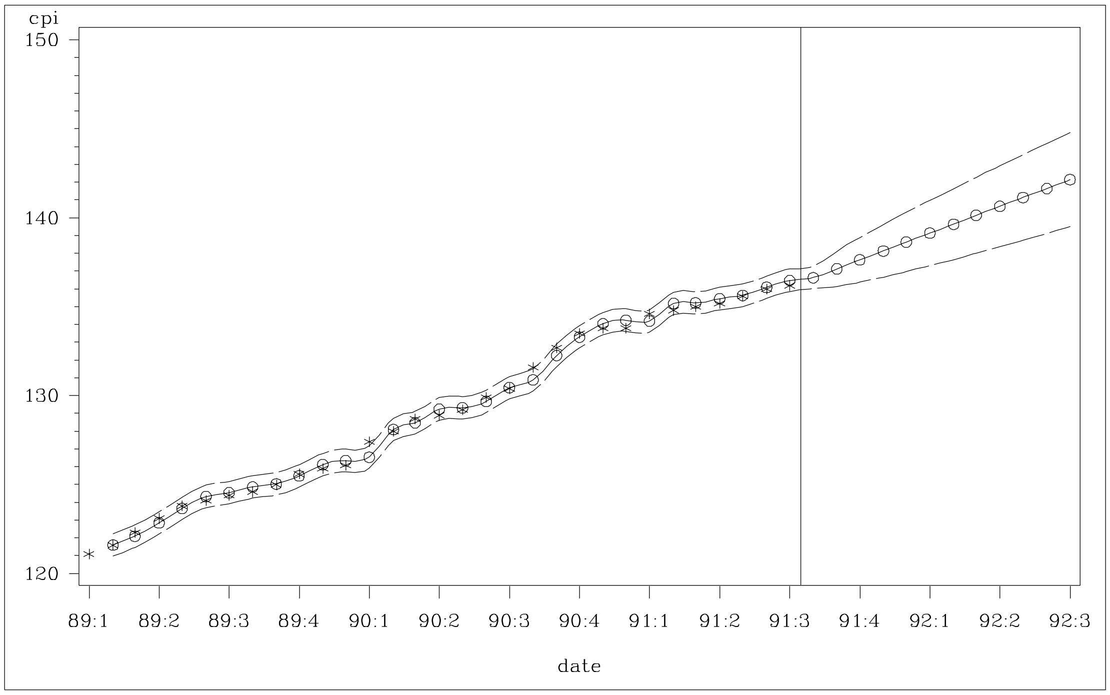


Figure 3.9. Plot of ARIMA Forecast

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 3.10.

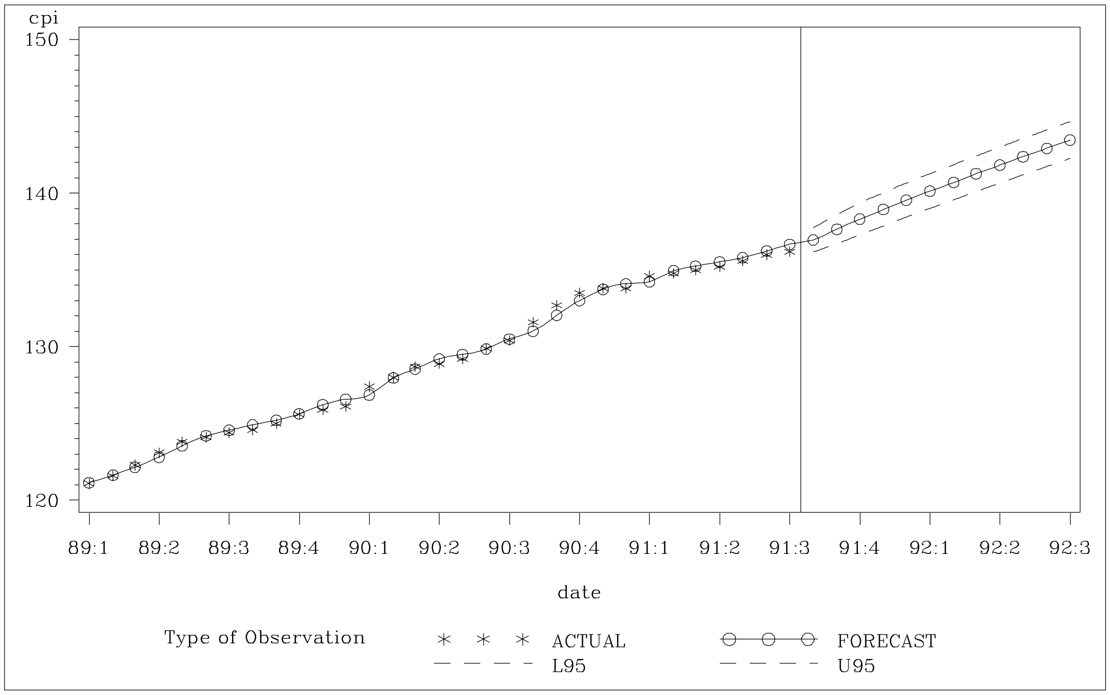


Figure 3.10. Plot of Forecast

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 3.11.

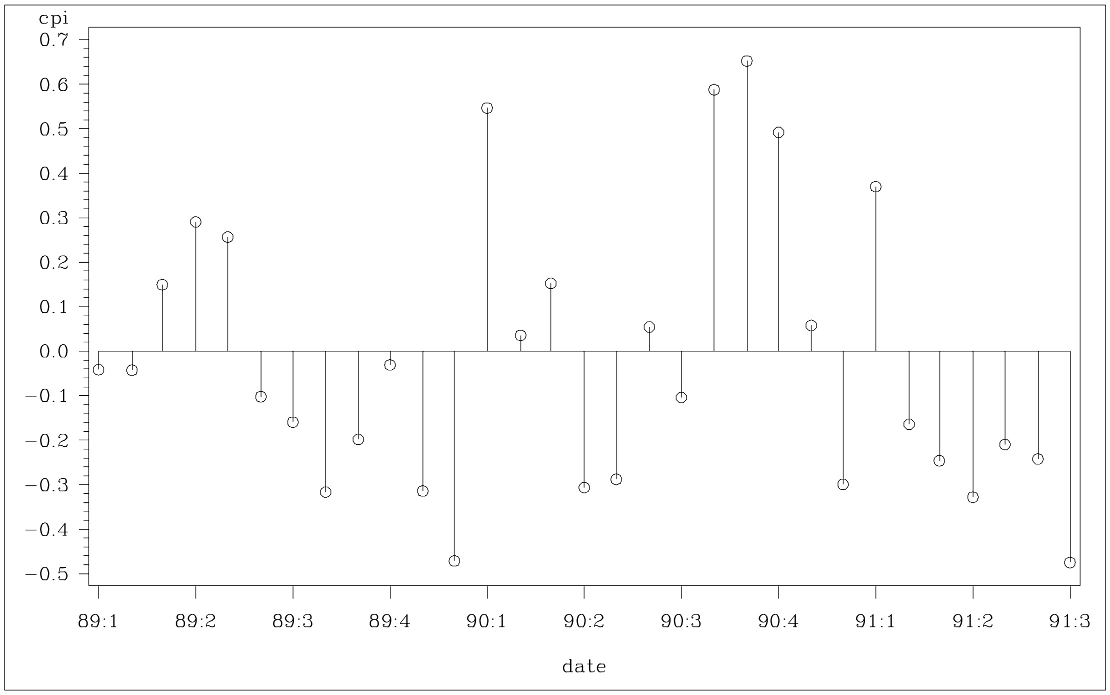


Figure 3.11. Plot of Residuals

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 3.12.

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

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

                                        date
```

Figure 3.12. 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 3.13.

```
                     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 3.13. 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 3.14.

```
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 3.14. 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 3.15.

```
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 3.15. 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 3.16.

```
   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 3.16. 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 3.17.

```
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 3.17. 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 3.18.

```
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 3.18. 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 3.19.

```
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 3.19. 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” (Chapter 3, *SAS/ETS User’s Guide*) 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 17, “The EXPAND Procedure,” (*SAS/ETS User’s Guide*) 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 3.20.

```
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 3.20. 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 21, "The MODEL Procedure," (*SAS/ETS User's Guide*) 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 17, "The EXPAND Procedure," (*SAS/ETS User's Guide*) 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 17, “The EXPAND Procedure,” (*SAS/ETS User’s Guide*) 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 3.21 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.

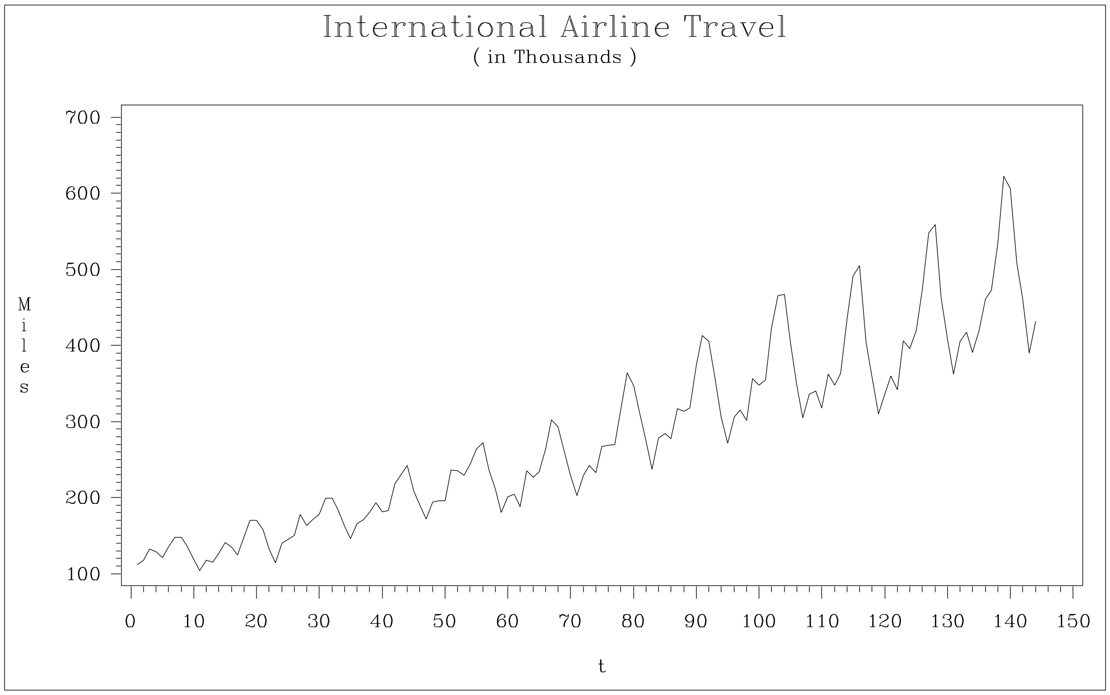


Figure 3.21. [Airline Series

The following statements compute the logarithms of the airline series:

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

Figure 3.22 shows a plot of the log transformed airline series. Notice that the log series has a linear trend and constant variance.

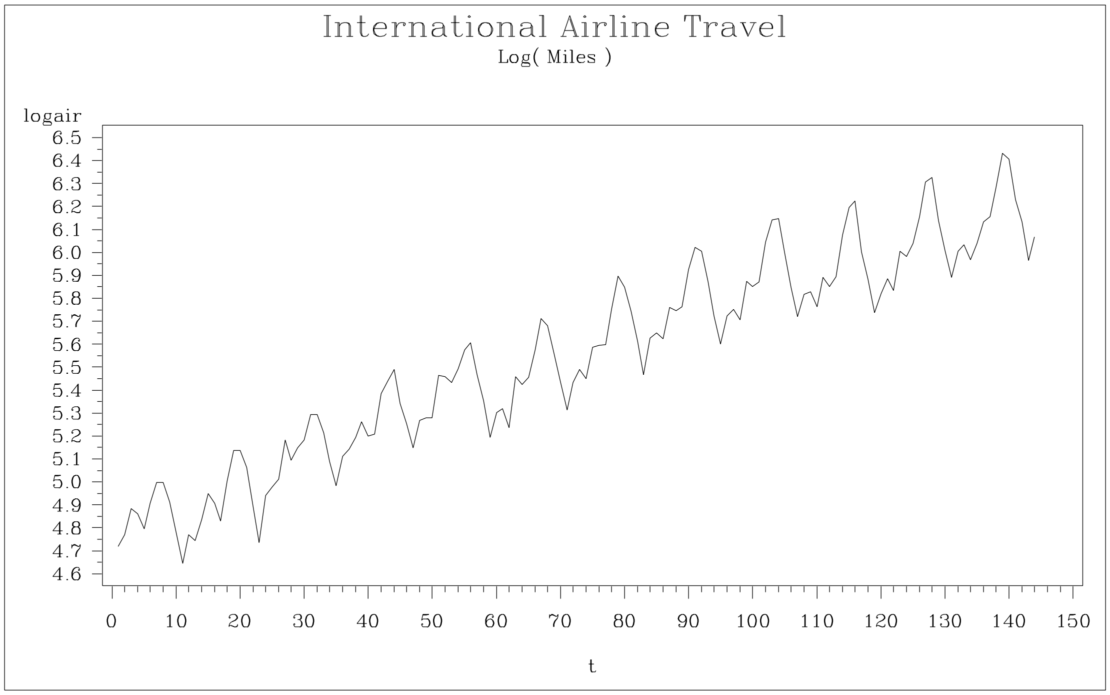


Figure 3.22. Log Airline Series

The %LOGTEST macro can help you decide if a log transformation is appropriate for a series. See Chapter 4, “SAS Macros and Functions,” (*SAS/ETS User’s Guide*) 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 (*SAS/ETS User’s Guide*) 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 -∞ to +∞.

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 17.1 (Chapter 17, *SAS/ETS User's Guide*) in Chapter 17 (*SAS/ETS User's Guide*) 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 3.23. (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 3.23. 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 3.24.

```
                   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 3.24. 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 3.25.

```
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 3.25. 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 17 (*SAS/ETS User's Guide*).

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 15, “The DATASOURCE Procedure,” (*SAS/ETS User’s Guide*) 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” (*SAS/ETS User’s Guide*) Chapter 6, “The SASEFAME Interface Engine” (*SAS/ETS User’s Guide*) and Chapter 7, “The SASEHAVR Interface Engine” (*SAS/ETS User’s Guide*) 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 3.26.

```
                             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 3.26. USPRICE Data Set

Chapter 4
Date Intervals, Formats, and Functions

Chapter Contents

Chapter 4
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 3, “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 3, “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.

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 periods, 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 4.1 shows examples of different kinds of interval specifications.

Table 4.1. 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 4.2 summarizes the SAS date and datetime informats available in the SAS System. See Chapter 3, “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 4.2 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 4.2 shows the width range allowed by the informat and the default width.

Table 4.2. 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 4.3 and Table 4.4. 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 3, "Working with Time Series Data," for a discussion of the use of date and datetime formats.

Date Formats

Table 4.3 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 4.3. 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 4.3. (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 4.3. (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 4.3. (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 4.4 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 4.4. 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 SAS date value given the Julian date in yyddd or yyyyddd format. For example, date = DATEJUL(99001); assigns the SAS date value '01JAN99'D to date, and date = DATEJUL(1999365); assigns the SAS date value '31DEC1999'D to date.

DATEPART(*datetime* **)**
returns the date part of a SAS datetime value as a date value.

DATETIME()
returns the current date and time of day as a SAS datetime value.

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.

HOLIDAY(*holiday, year* **)**
returns a SAS date value for the holiday and year specified. Valid values for holiday are 'EASTER', 'THANKSGIVING', 'BOXING', 'CANADA', 'CHRISTMAS', 'COLUMBUS', 'FATHERS', 'HALLOWEEN', 'USINDEPENDENCE', 'LABOR', 'MEMORIAL', 'MOTHERS', 'NEWYEAR', 'THANKSGIVINGCANADA', 'VALENTINES', 'VETERANS', 'VETERANSUSG', 'VETERANSUSPS', 'VICTORIA', and 'CANADAOBSERVED'. For example, Easter2000 = HOLIDAY('EASTER', 2000);

HOUR(*datetime* **)**
returns the hour from a SAS datetime or time value.

INTCINDEX(*'interval', value* **)**
returns the index of the seasonal cycle given an interval and an appropriate SAS date, datetime, or time value. For example, the seasonal cycle for interval='DAY' is 'WEEK', so INTCINDEX('DAY','01SEP78'D); returns 35 since September 1, 1978, is the sixth day of the 35th week of the year.

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 interval of the seasonal cycle, given a date, time, or datetime interval. For example, INTCYCLE('MONTH') returns 'YEAR' since the months January, February, ..., December constitute a yearly cycle. INTCYCLE('DAY') returns 'WEEK' since Sunday, Monday, ..., Saturday is a weekly cycle.

INTFMT(*'interval', 'size'* **)**
returns a recommended format, given a date, time, or datetime interval for displaying the time ID values associated with a time series of the given interval. The valid values of size: 'long', 'l', 'short', 's' specify whether the user prefers to use a 2-digit or 4-digit year when the format refers to a SAS date value.

INTINDEX(*'interval', value* **)**
returns the seasonal index, given a date, time, or datetime interval and an appropriate date, time, or datetime value. The seasonal index is a number representing the position of the date, time, or datetime value in the seasonal cycle of the specified interval. For example, INTINDEX('MONTH','01DEC2000'D); returns 12 since monthly data is yearly periodic and DECEMBER is the 12th month of the year. However, INTINDEX('DAY,'01DEC2000'D); returns 6 since daily data is weekly periodic and December 01, 2000 is a Friday, the sixth day of the week. To correctly identify the seasonal index, the interval format should agree with the date, time, or datetime value. For example, INTINDEX('DTMONTH','01DEC2000'D); and INTINDEX('MONTH','01DEC2000:00:00:00'DT); do not return the expected value of 12. However, both INTINDEX('MONTH','01DEC2000'D); and INTINDEX('DTMONTH','01DEC2000:00:00:00'DT); return the expected value of 12.

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.

INTSEAS(*'interval'* **)**
returns the length of the seasonal cycle, given a date, time, or datetime interval. The length of a seasonal cycle is the number of intervals in a seasonal cycle. For example, when the interval for a time series is described as monthly, many procedures use the option INTERVAL=MONTH. Each observation in the data then corresponds to a particular month. Monthly data is considered to be periodic for a one-year period. There are 12 months in one year, so the number of intervals (months) in a seasonal cycle (year) is 12. For quarterly data, there are 4 quarters in one year, so the number of intervals in a seasonal cycle is 4. The periodicity is not always one year. For example, INTERVAL=DAY is considered to have a period of one week, and since there are 7 days in a week, the number of intervals in a seasonal cycle is 7.

INTTEST(*'interval'* **)**
returns 1 if the interval name is a valid interval, 0 otherwise. For example, VALID = INTTEST('month'); should set VALID to 1, while VALID = INTTEST('NotAnInterval'); should set VALID to 0. The INTTEST function can be useful in verifying which values of multiplier *n* and the shift index *s* are valid in constructing an interval name.

JULDATE(*date* **)**
returns the Julian date from a SAS date value. The format of the Julian date is either yyddd or yyyyddd depending on the value of the system option YEARCUTOFF=.

For example, using the default system option values, JULDATE(’31DEC1999’D); returns 99365, while JULDATE(’31DEC1899’D); returns 1899365.

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.

NWKDOM(*n, weekday, month, year* **)**
returns a SAS date value for the nth weekday of the month and year specified. For example, Thanksgiving is always the fourth Thursday in November. Thanks2000 = NWKDOM(4, 5, 11, 2000); returns the SAS date value for Thanksgiving in the year 2000. The last weekday of a month may be specified using $n = 5$. Memorial Day is the last Monday in May. Memorial2002 = NWKDOM(5, 2, 5, 2002); returns the SAS date value for Memorial Day in 2002.

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 5
Using the Output Delivery System

Chapter Contents

Chapter 5
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 122 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 5.1. Alternatively, you can specify the LISTING option, which writes the information, interleaved with the procedure output, to the SAS listing (see Example 5.1).

```
   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 5.1. Partial Contents of the SAS Log: Result of the ODS TRACE Statement

Figure 5.1 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 5.2 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 5.2 displays the output hierarchy when the preceding statements are executed.

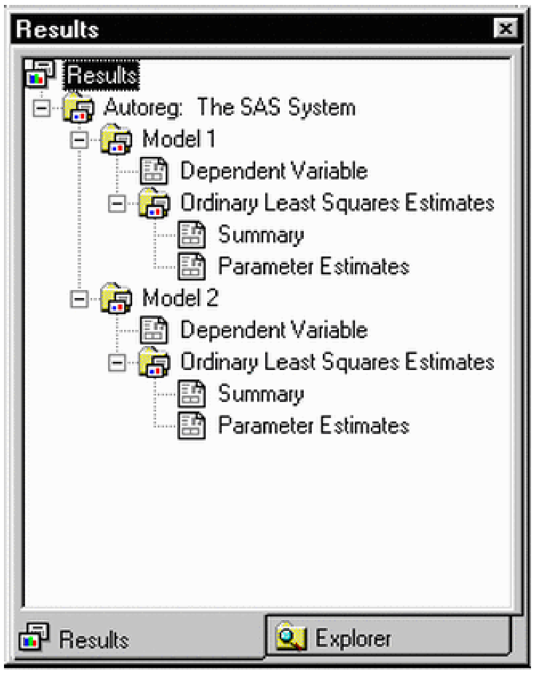


Figure 5.2. The Results Window from the SAS Explorer

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 "etsug.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 5.1. 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 5.1.1 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;
```

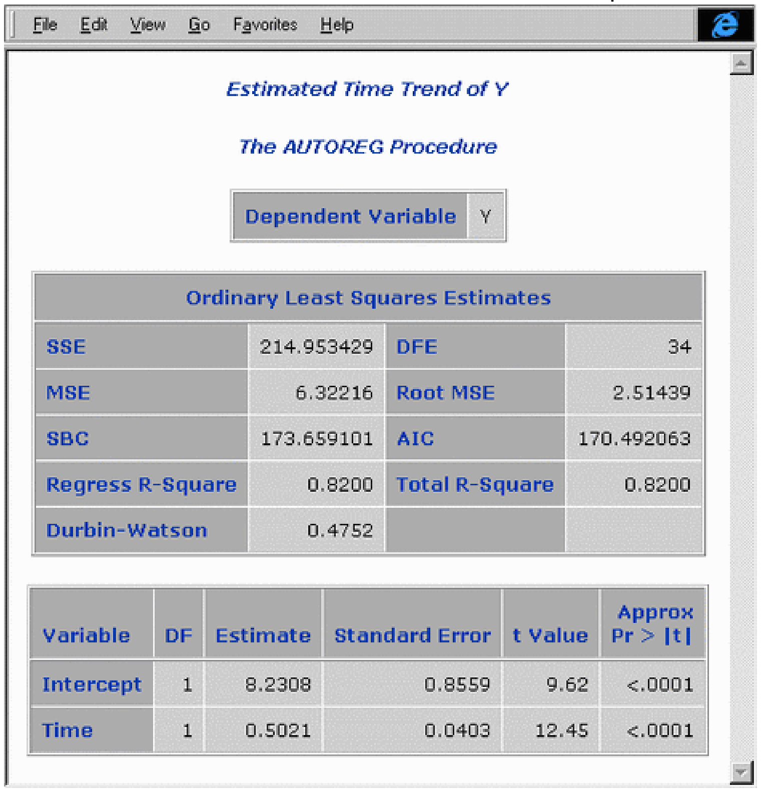
If you do not close the HTML destination, your HTML file may contain no output, or you may experience other unexpected results.

Output 5.1.2 displays the file 'trend.htm', which is specified in the preceding ODS HTML statement.

Output 5.1.1. 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 5.1.2. 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 5.2. 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.

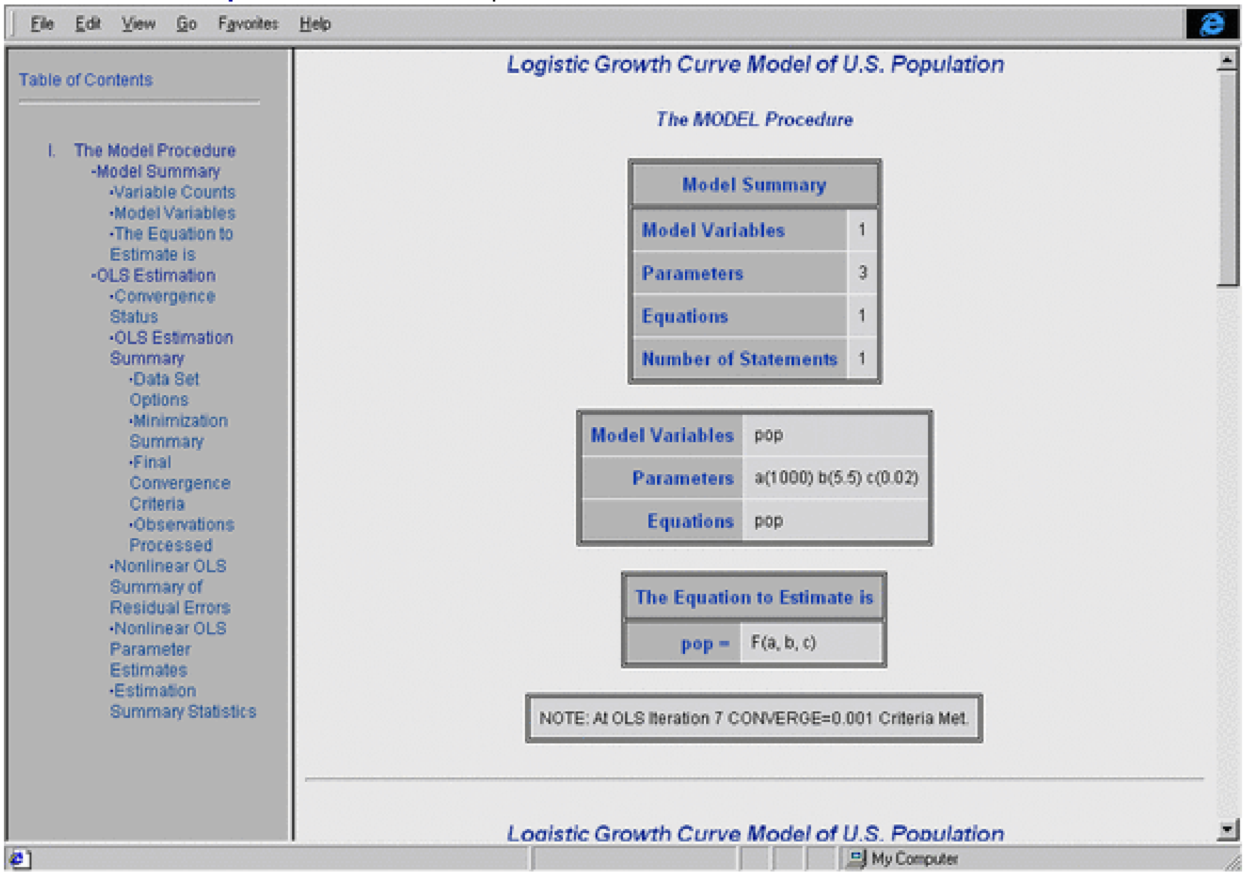
```
data uspop;
   input pop :6.3 @@;
   retain year 1780;
   year=year+10;
   label pop='U.S. Population in Millions';
   datalines;
3929   5308   7239   9638   12866  17069  23191  31443
39818  50155  62947  75994  91972  105710 122775 131669
151325 179323 203211 226542 248710
;

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 5.2.1). 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 5.2.1.

Output 5.2.1. HTML Output from the MODEL Procedure



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 5.2.1 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 5.3. 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 5.3.1. 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 5.3.1, 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 119.

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 5.4. 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 5.4.1 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 5.4.1. 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 5.4.2.

Output 5.4.2. 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 119.

```
ods show;
quit;
ods show;
```

The results of the statements are displayed in Output 5.4.3. Before the MODEL procedure terminates, the ODS selection list includes only the two tables, "ResidSummary" and "ParameterEstimates."

Output 5.4.3. 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 5.4.4.

Output 5.4.4. 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 5.5. 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 5.5.1. 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 5.5.1. Note that you can alternatively specify that the information be interleaved with the procedure output in the SAS listing (see Example 5.3).

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 5.5.2.

```
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 5.5.2. 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 6
Statistical Graphics Using ODS (Experimental)

Chapter Contents

Chapter 6 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 178. 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 152.

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.” (*SAS/ETS User’s Guide*) 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 145, 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 152, the section "Graphics Image Files" on page 162, and the section "Examples" beginning on page 181.

If you need to customize a graph by modifying its template, read the section "Customizing Graphics with Templates" on page 167 and the series of examples beginning on page 195.

If you need to customize a style definition read the section "Styles for Graphics" on page 174 and the series of examples beginning on page 206.

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 21, "The MODEL Procedure." (*SAS/ETS User's Guide*) 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 152.

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 6.1 through Figure 6.8, respectively.

The ODS GRAPHICS OFF statement disables ODS Graphics, and the ODS HTML CLOSE statement closes the HTML destination.

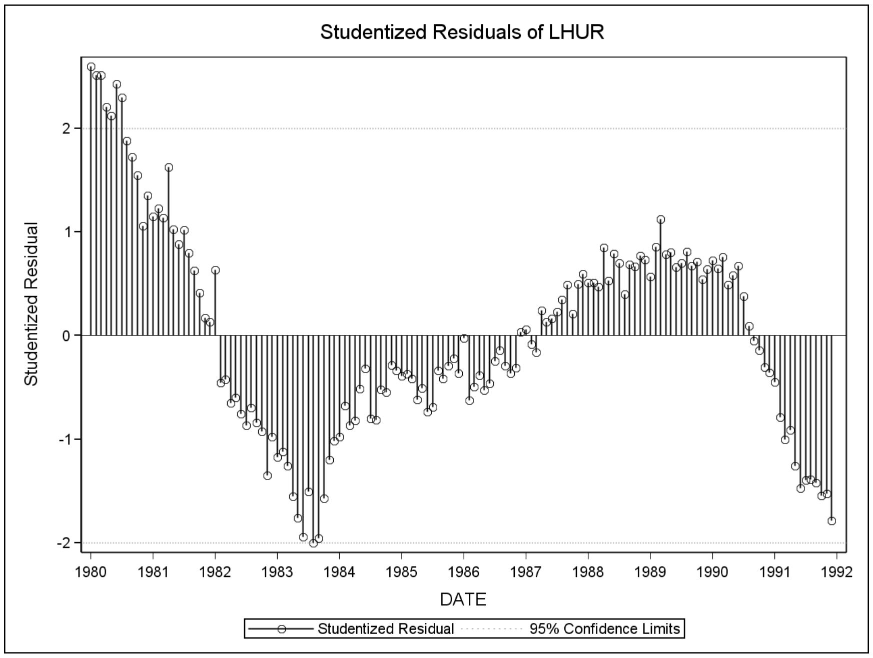


Figure 6.1. Studentized Residuals

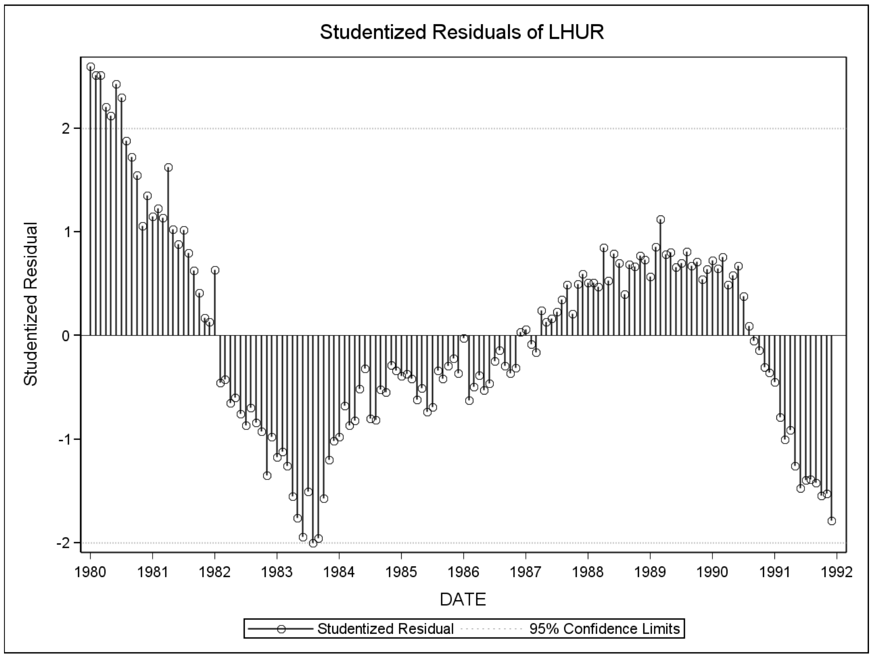


Figure 6.2. Cook's D for Residuals

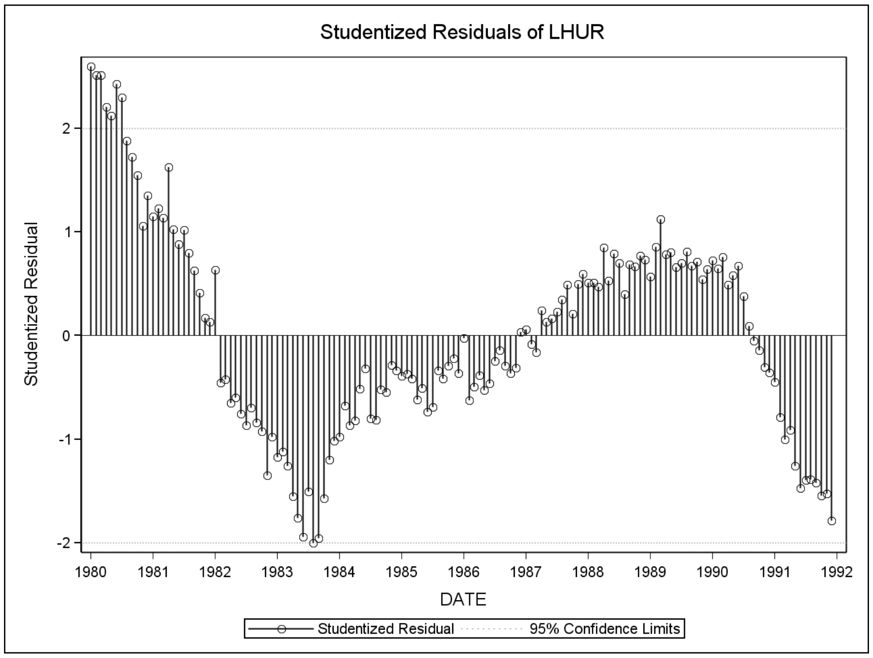


Figure 6.3. Predicted and Actual Values

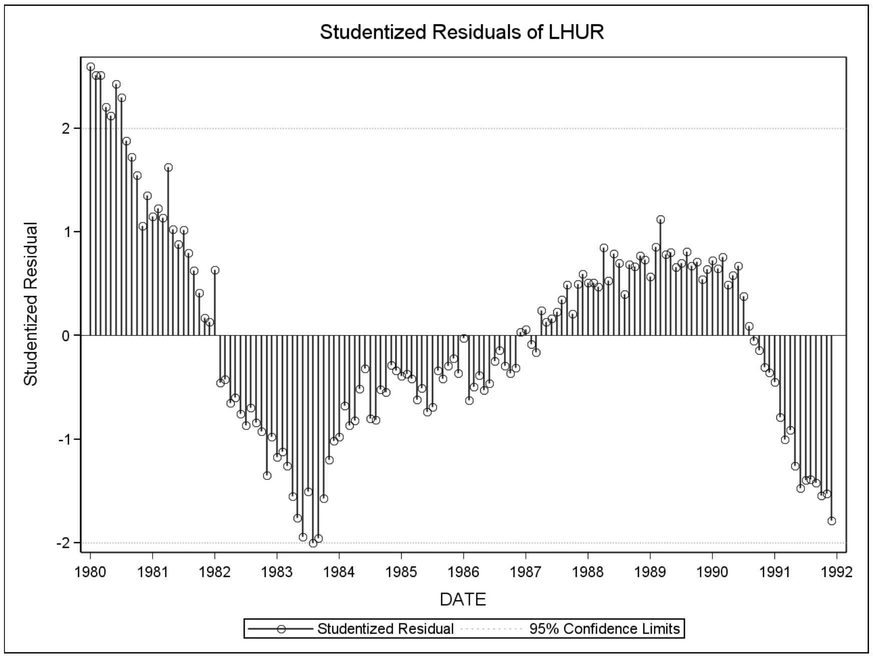


Figure 6.4. Autocorrelation of Residuals

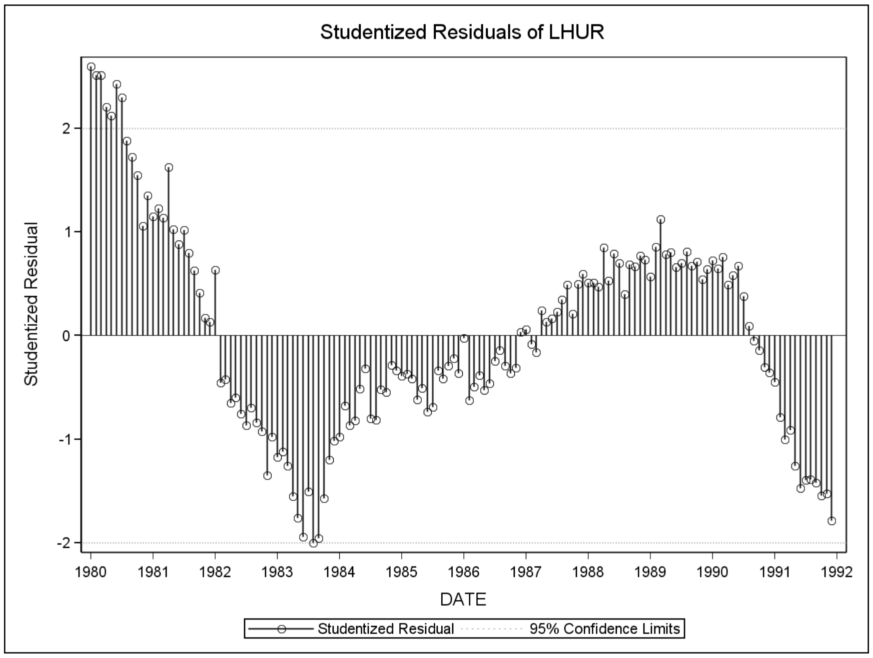


Figure 6.5. Partial Autocorrelation of Residuals

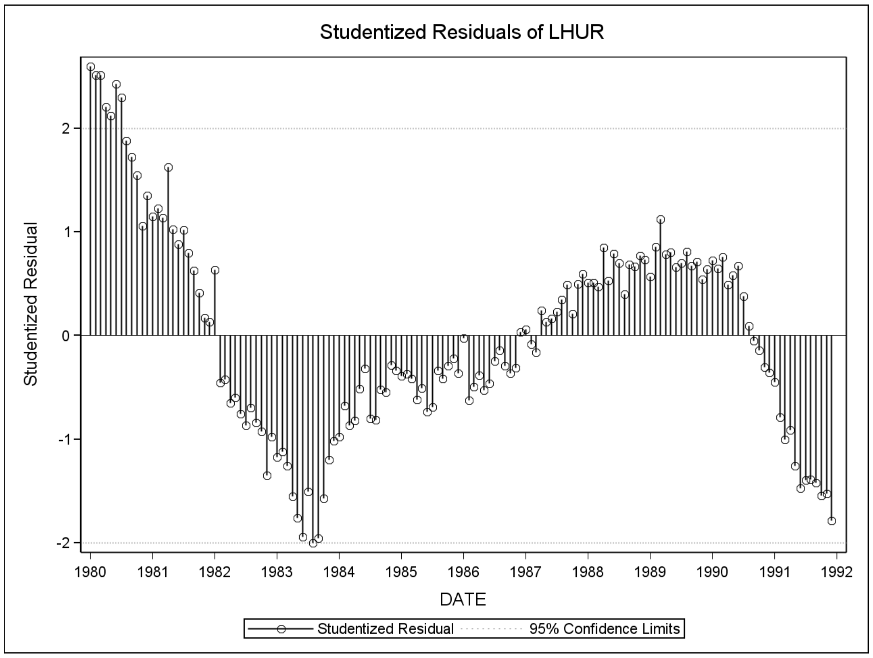


Figure 6.6. Inverse Autocorrelation of Residuals

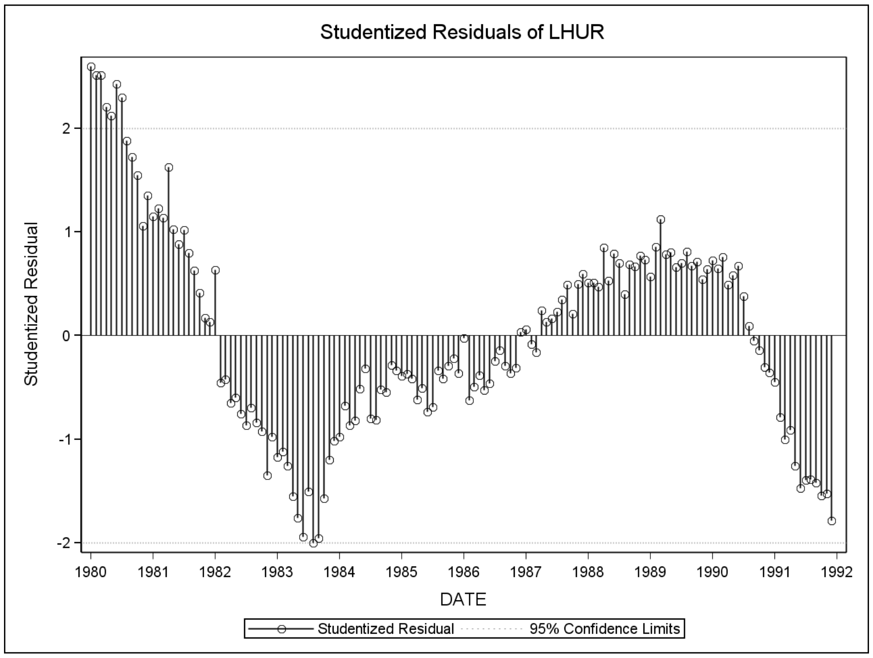


Figure 6.7. QQ Plot of Residuals

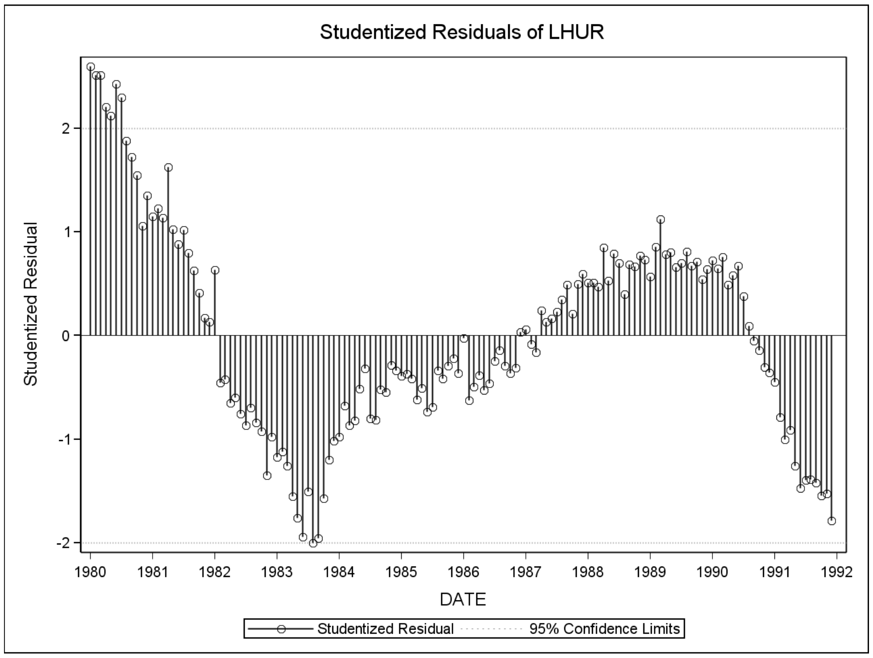


Figure 6.8. Histogram of Residuals

For more information about ODS Graphics available in the MODEL procedure, see the "ODS Graphics" (Chapter 21, *SAS/ETS User's Guide*) section on page 1333 in Chapter 21, "The MODEL Procedure." (*SAS/ETS User's Guide*)

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 6.9 and Figure 6.10, respectively. This graphical display are requested by specifying the ODS GRAPHICS statement prior to the procedure statements, and the experimental PLOTS= (Chapter 30, *SAS/ETS User's Guide*) options in the CYCLE and FORECAST statements. For more information about the graphics available in the UCM procedure, see the "ODS Graphics" (Chapter 30, *SAS/ETS User's Guide*) section on page 1892 in Chapter 30, "The UCM Procedure." (*SAS/ETS User's Guide*)

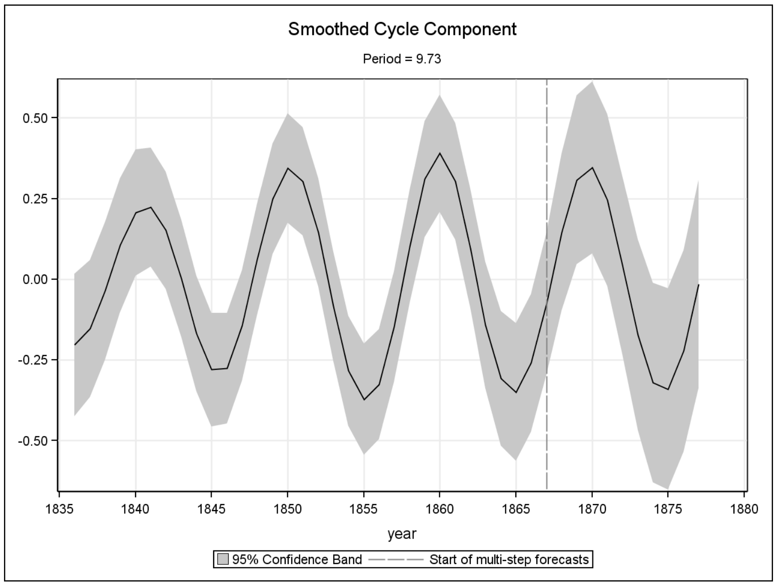


Figure 6.9. Smoothed Cycle Component Plot

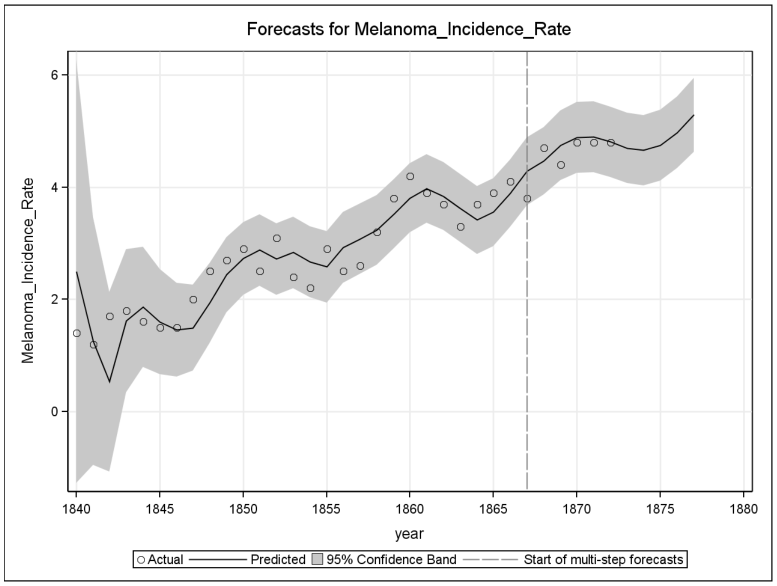


Figure 6.10. Forecasts Plot

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 6.1.

Table 6.1. 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 6.11.

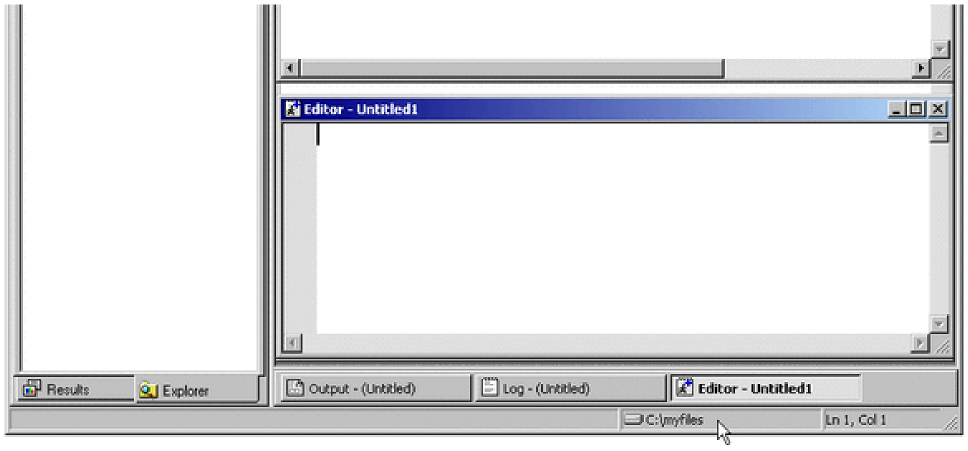


Figure 6.11. Current Folder (Right Bottom)

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
```

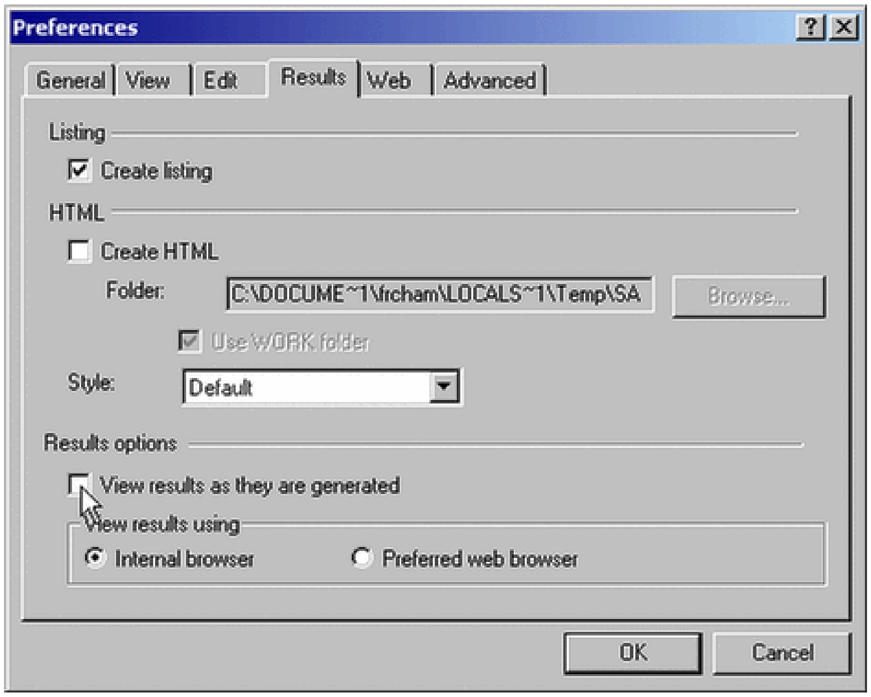
The default file names for each destination are specified in the SAS Registry. For more information, refer to the SAS Companion for your operating system.

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 6.12.



Preferences

General | View | Edit | Results | Web | Advanced

Listing
- [✓] Create listing

HTML
- [ ] Create HTML
- Folder: C:\DOCUME~1\frcham\LOCALS~1\Temp\SA   Browse...
- [✓] Use WORK folder
- Style: Default

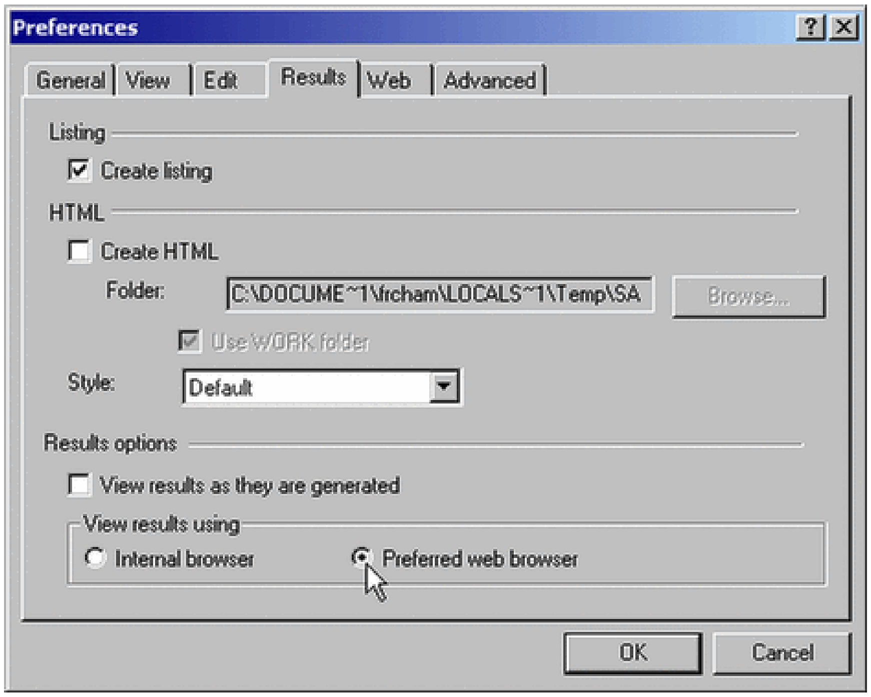
Results options
- [ ] View results as they are generated
- View results using: [✓ Internal browser] Preferred web browser

OK   Cancel

Figure 6.12. 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 6.13. Your results will then be displayed in the default viewer that is configured in your computer for the corresponding destination.



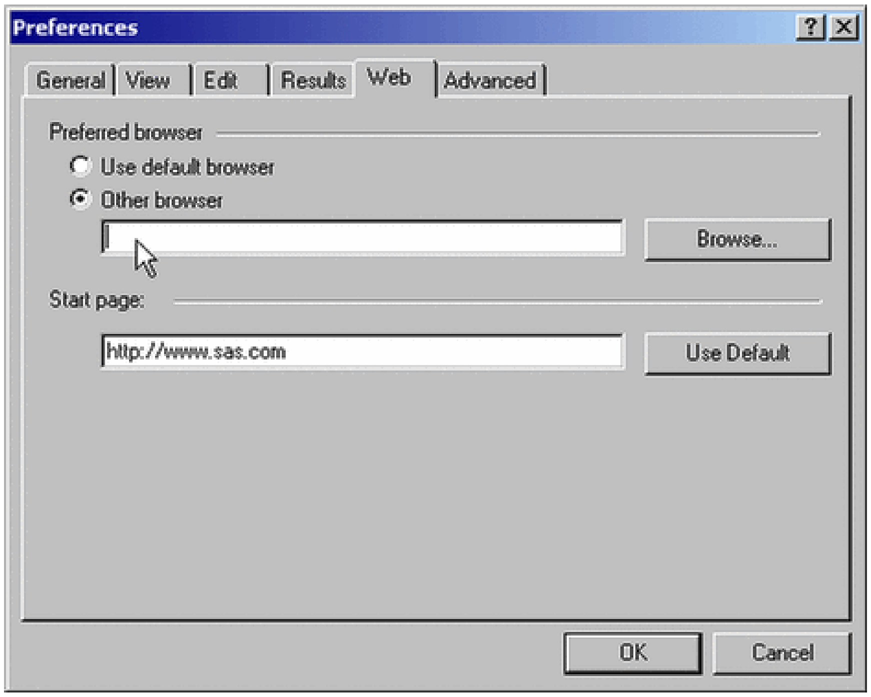
Listing
[✓] Create listing

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 6.13. 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 6.14.



Preferred browser
( ) Use default browser
(•) Other browser
Start page:
http://www.sas.com

Figure 6.14. 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" (Chapter 8, *SAS/ETS User's Guide*) section on page 363 in Chapter 8, "Using the Output Delivery System" (*SAS/ETS User's Guide*)). 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" (Chapter 21, *SAS/ETS User's Guide*) section on page 1333 in Chapter 21, "The MODEL Procedure." (*SAS/ETS User's Guide*)
- 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" (Chapter 8, *SAS/ETS User's Guide*) on page 366 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 157 for an example, and the "Using the Output Delivery System" (Chapter 8, *SAS/ETS User's Guide*) section on page 363 for more information.

Note that the graph name is not the same as the name of the file containing the graph (see page 164).

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" (Chapter 8, *SAS/ETS User's Guide*) on page 363 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 150.

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 6.15 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 6.15, 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 6.15. 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;
```

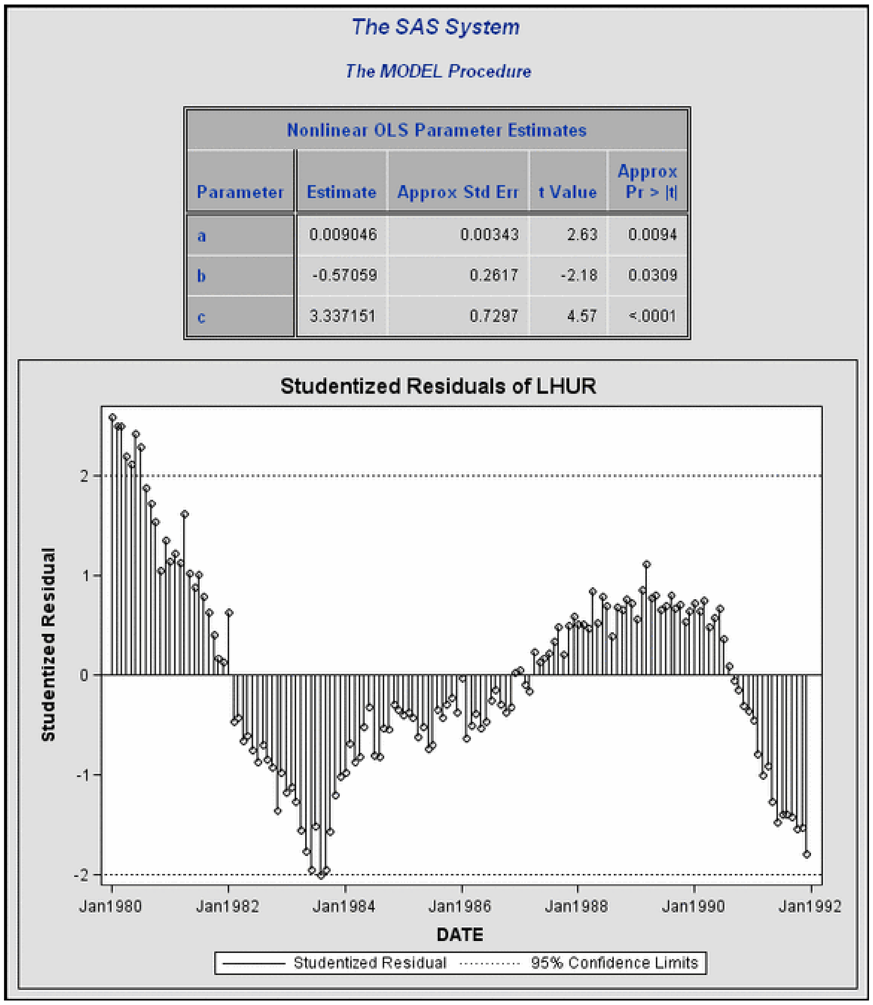
Any SAS-supplied or user-defined style can be used for ODS Graphics. However, of the SAS-supplied styles for SAS 9.1, four are specifically designed and recommended for use with ODS Graphics:

- Analysis
- Default
- Journal
- Statistical

Figure 6.16 and Figure 6.17 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 6.11.

For more information about styles for ODS Graphics, see the section "Styles for Graphics" on page 174 or refer to the "ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)" at
http://support.sas.com/documentation/onlinedoc/base/.

[†]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.

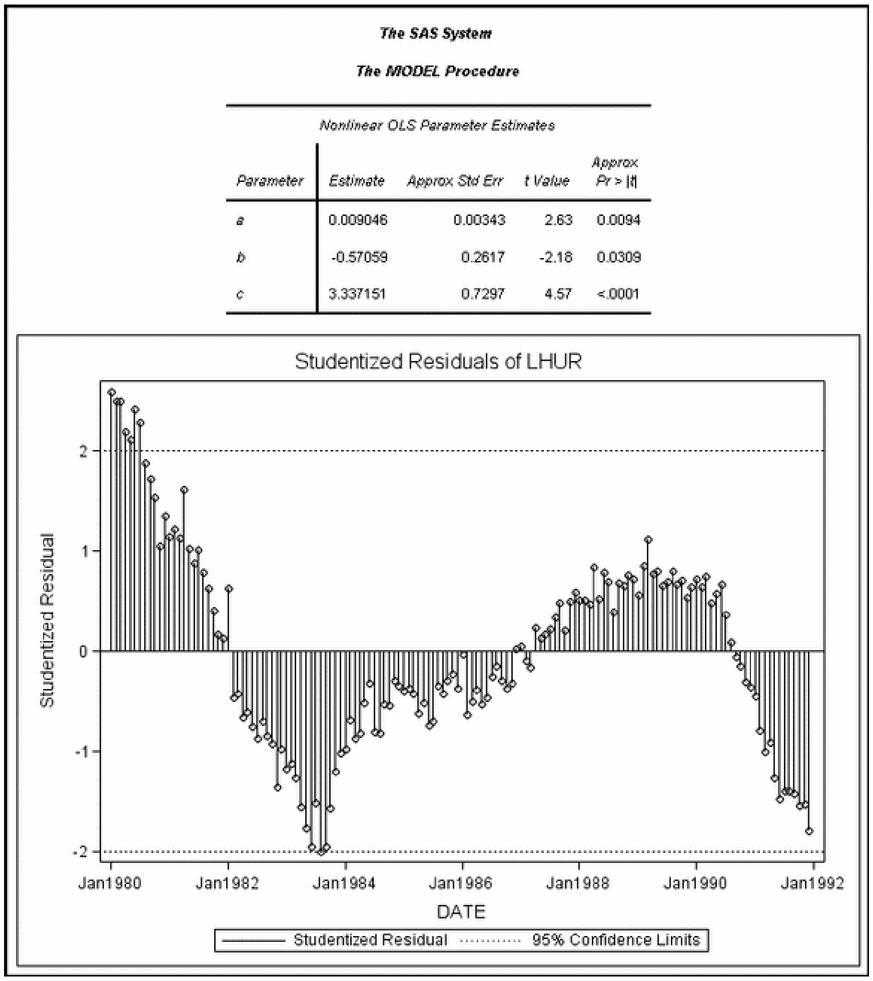


The SAS System

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 6.16. 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 6.17. 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 6.3.

Table 6.2 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 6.2. 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 156). 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 150 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 179.

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 154), 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 153). 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 6.4 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 160.

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 6.7 and the section "Using the Output Delivery System" (Chapter 8, *SAS/ETS User's Guide*) on page 363. 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 6.18. 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 6.7.

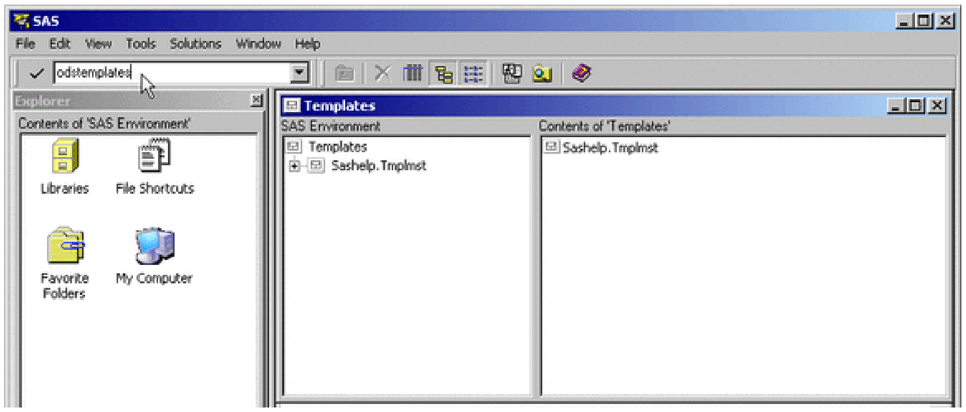


Figure 6.18. Requesting the Templates Window in the Command Line

- 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 6.7.
- 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 172.

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 6.7.
- 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 6.19.

```
Current ODS PATH list is:

1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

Figure 6.19. 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 6.7 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” (Chapter 29, *SAS/ETS User’s Guide*) in Chapter 29, “The TIMESERIES Procedure” (*SAS/ETS User’s Guide*)).

```
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 174.

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 6.13 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 6.11 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 6.12 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 168. 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.

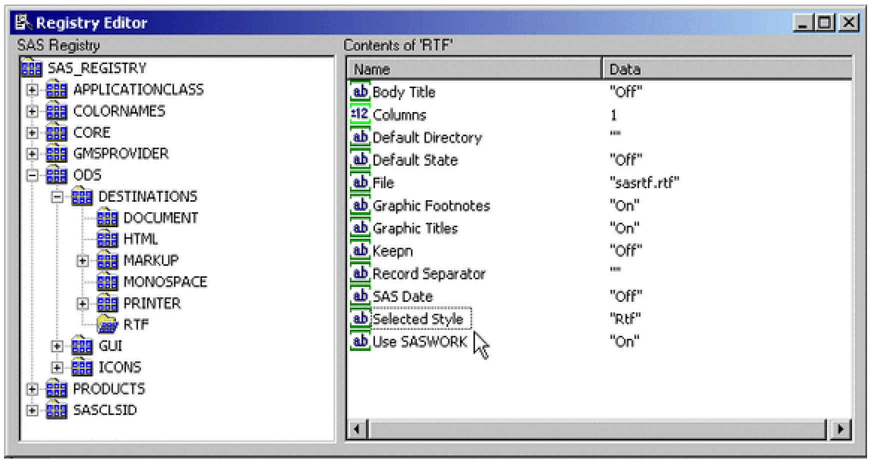
```
proc template;
   source Styles.Journal;
run;
```

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 175), 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 6.20, 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.



Contents of 'RTF'

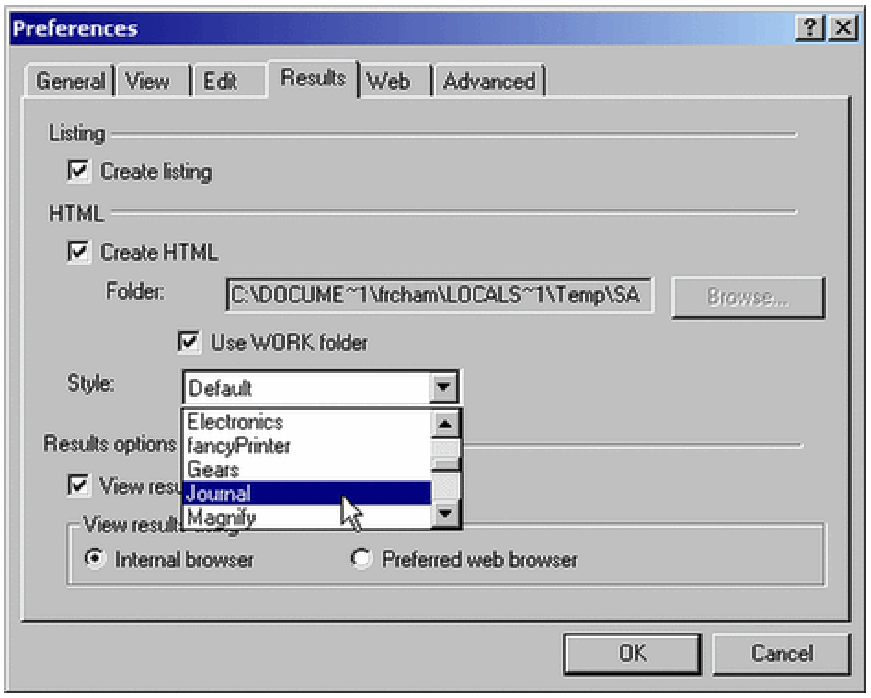
Name	Data
Body Title	"Off"
Columns	1
Default Directory	""
Default State	"Off"
File	"sasrtf.rtf"
Graphic Footnotes	"On"
Graphic Titles	"On"
Keepn	"Off"
Record Separator	""
SAS Date	"Off"
Selected Style	"Rtf"
Use SASWORK	"On"

Figure 6.20. 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 6.21.



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 6.21. 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 145. 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 6.3 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 6.2. If the number of observations in the data set exceeds 500, the image map is not generated.

Table 6.3. 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 6.2.

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 6.1. Selecting and Excluding Graphs

This example illustrates how to select and exclude ODS graphs from your output.

The “Getting Started” example on page 145 uses the MODEL procedure to produce the plots shown in Figure 6.1 through Figure 6.8.

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 6.1.1.

```
   ods trace on;
p
   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;
```

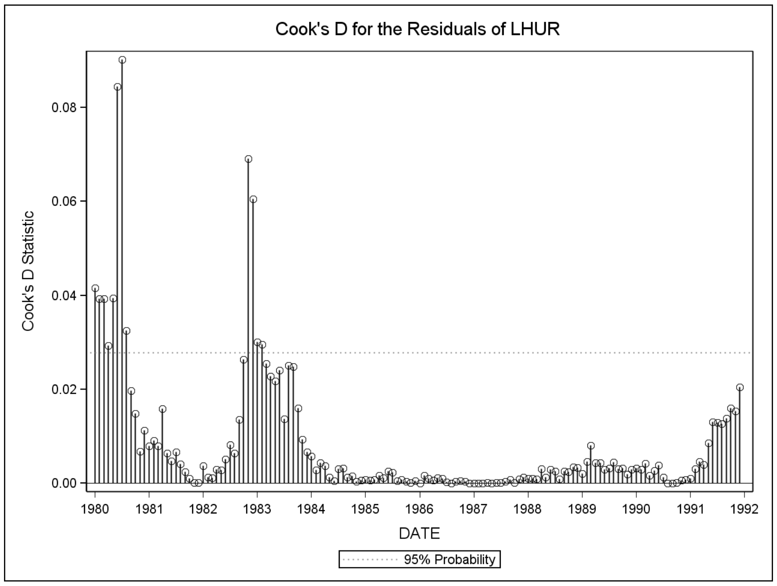
Output 6.1.1. Partial ODS Trace Record in SAS Log

```
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 6.1.2.

```
   ods html;
   ods graphics on;
p

   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 6.1.2. 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 157 for further information.

A sample program named odsgr01.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 6.2. 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 29.3 (Chapter 29, *SAS/ETS User's Guide*) of Chapter 29, "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.

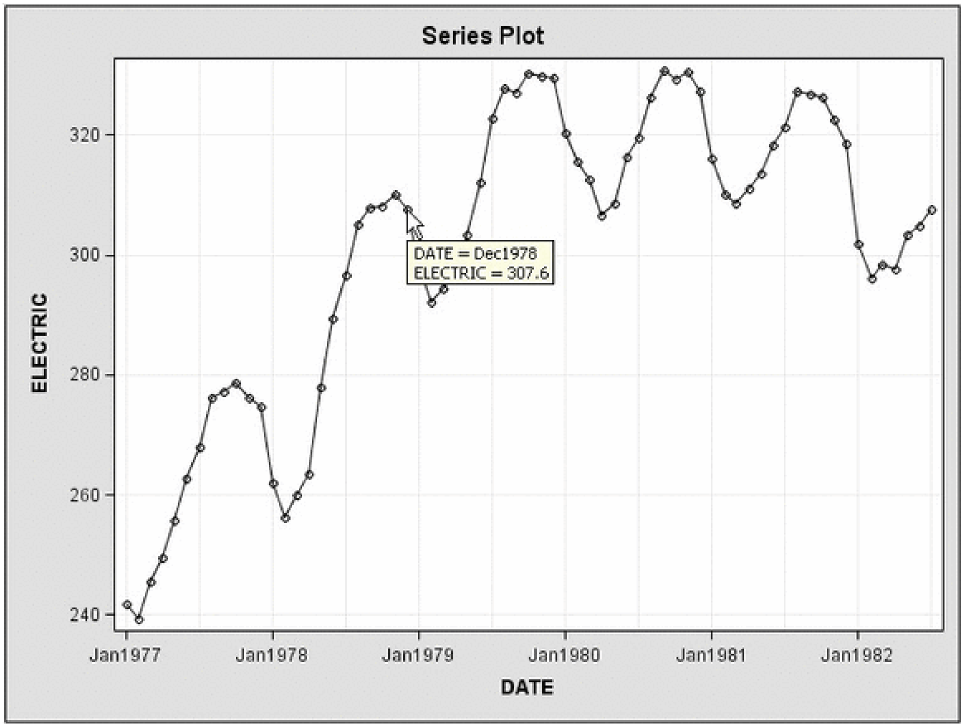
The following statements request a plot of the series of electrical workers using the PLOT (Chapter 29, *SAS/ETS User's Guide*) option in the PROC TIMESERIES statement.

```
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 6.2.1 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 6.2.1. 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 6.3. Creating Graphs for a Presentation

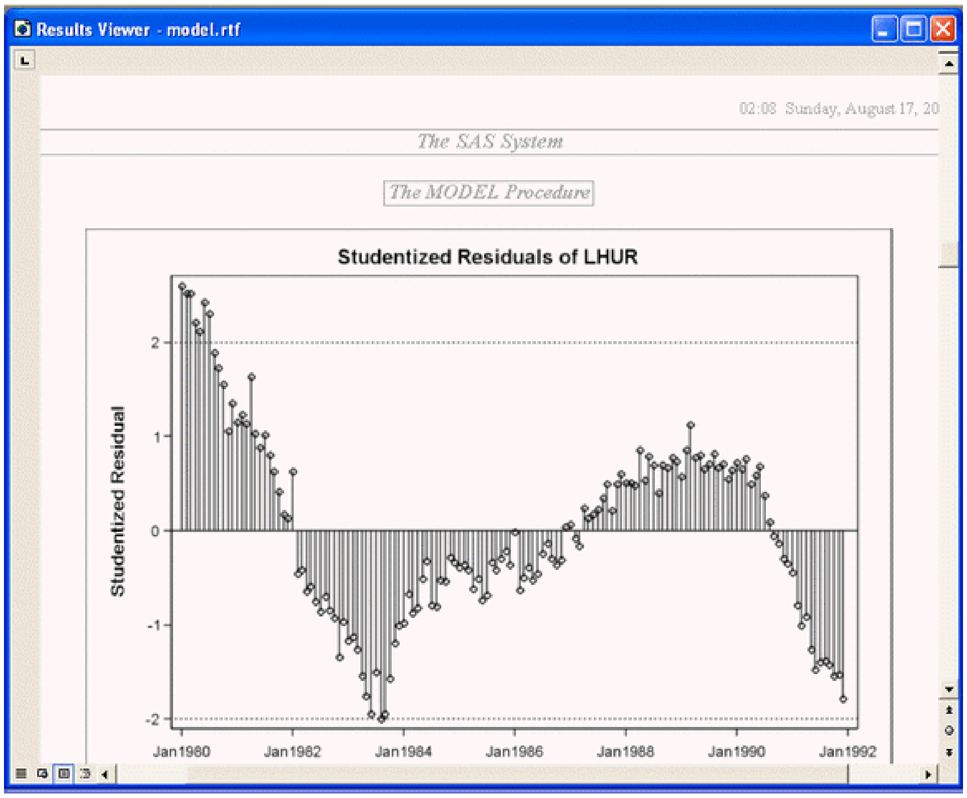
The RTF destination provides the easiest way to create ODS graphs for inclusion in a document or presentation. You can specify the ODS RTF statement to create a file that is easily imported into a word processor (such as Microsoft Word or WordPerfect) or a presentation (such as Microsoft PowerPoint).

In this example, the following statements request that the output of Example 6.1 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 6.3.1.

Output 6.3.1. 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 6.4. 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.

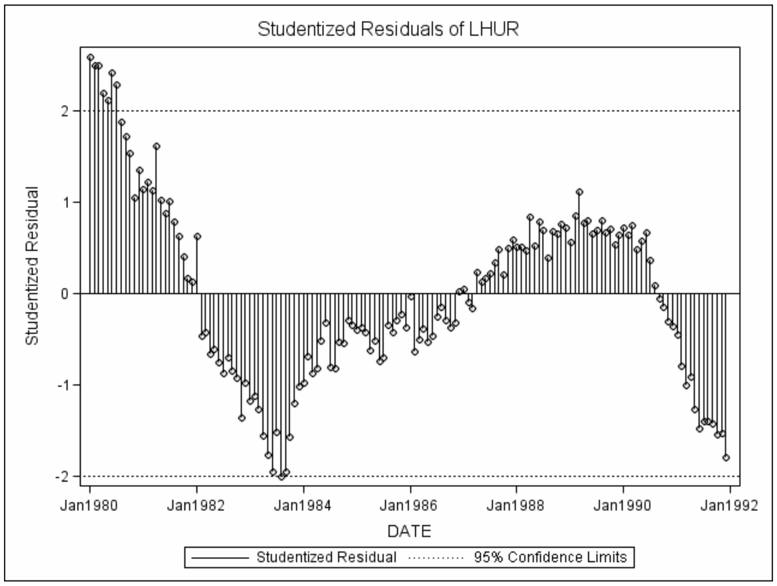
The following statements specify a LATEX destination[‡] for the output in Example 6.3 with the "Journal" style.

```
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 6.4.1 is saved by default in a file named StudentResidualPlot0.ps. See page 164 for details about how graphics image files are named.

[‡]The LATEX destination in ODS is experimental in SAS 9.1.

Output 6.4.1. 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 165 for further information.

A sample program named odsgr04.sas is available for this example in the SAS Sample Library for SAS/ETS software.

Example 6.5. 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 150, 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 6.6. 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 6.1. 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 156 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.

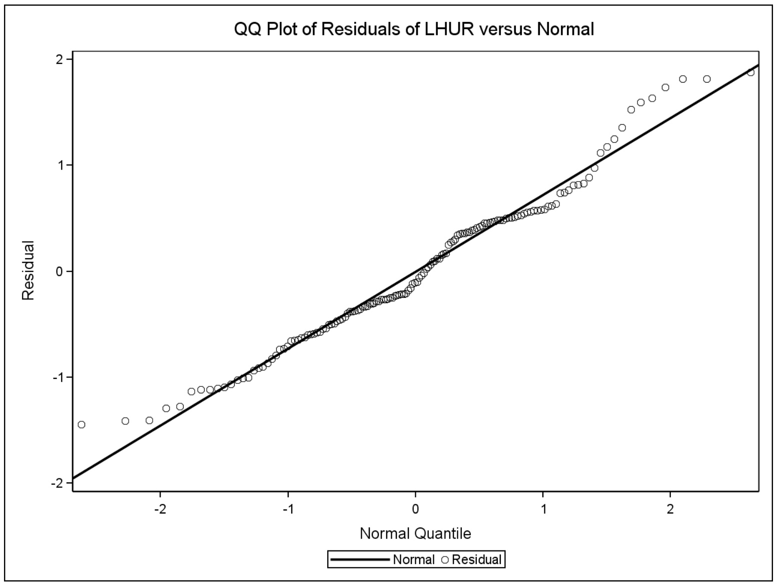
The following statements specify an HTML destination and display the residual Q-Q plot using the REPLAY statement in PROC DOCUMENT.

```
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 6.6.1.

Output 6.6.1. 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 6.6.2.

Output 6.6.2. 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#2                                                         Dir
      6 \Model#2\ModSum#1                                                Dir
      7 \Model#2\ModSum#1\ModSummary#1                                   Table
      8 \Model#2\ModSum#1\ModVars#1                                      Tree
      9 \Model#2\ModSum#1\Equations#1                                    Tree
     10 \Model#2\OLS#1                                                   Dir
     11 \Model#2\OLS#1\ConvergenceStatus#1                               Table
     12 \Model#2\OLS#1\EstSum#1                                          Dir
     13 \Model#2\OLS#1\EstSum#1\DatasetOptions#1                         Table
     14 \Model#2\OLS#1\EstSum#1\MinSummary#1                             Table
     15 \Model#2\OLS#1\EstSum#1\ConvCrit#1                               Table
     16 \Model#2\OLS#1\EstSum#1\ObsUsed#1                                Table
     17 \Model#2\OLS#1\ResidSummary#1                                    Table
     18 \Model#2\OLS#1\ParameterEstimates#1                              Table
     19 \Model#2\OLS#1\EstSummaryStats#1                                 Table
     20 \Model#2\OLS#1\StudentResidualPlot#1                             Graph
     21 \Model#2\OLS#1\CooksD#1                                          Graph
     22 \Model#2\OLS#1\ActualByPredicted#1                               Graph
     23 \Model#2\OLS#1\ACFPlot#1                                         Graph
     24 \Model#2\OLS#1\PACFPlot#1                                        Graph
     25 \Model#2\OLS#1\IACFPlot#1                                        Graph
     26 \Model#2\OLS#1\QQPlot#1                                          Graph
     27 \Model#2\OLS#1\ResidualHistogram#1                               Graph
```

The document path of the "QQPlot" entry in lhurDoc, as shown in Output 6.6.2, 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 6.7. Customizing Graph Titles and Axes Labels

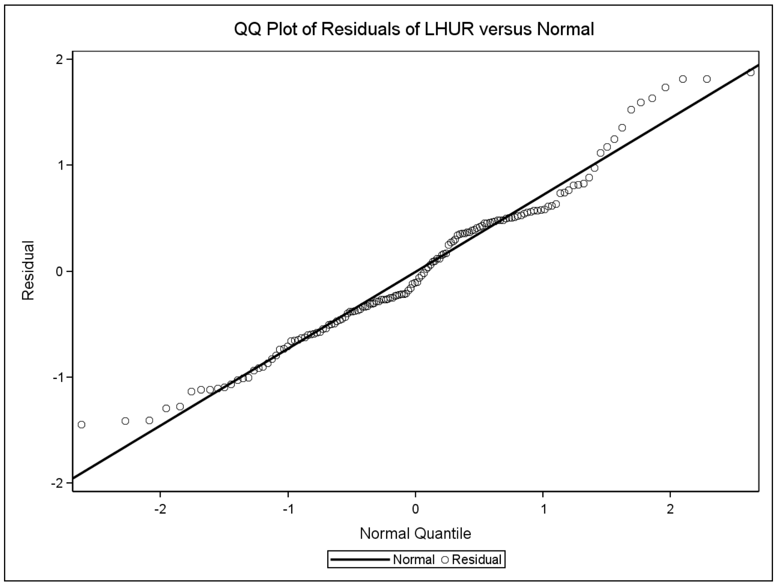
This example shows how to use PROC TEMPLATE to customize the appearance and content of an ODS graph. It illustrates the discussion in the section "Customizing Graphics with Templates" on page 167 in the context of changing the default title and y-axis label for a Q-Q plot created with the MODEL procedure.

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 6.1.

```
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 6.7.1.

Output 6.7.1. 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 6.7.2.

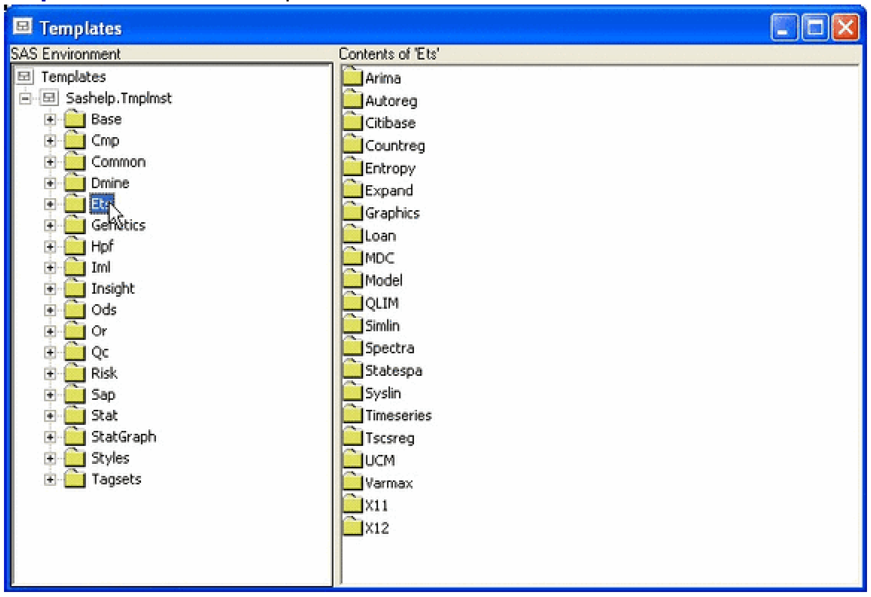
Output 6.7.2. Trace Record for Q-Q Plot

```
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 6.7.2, 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 167).

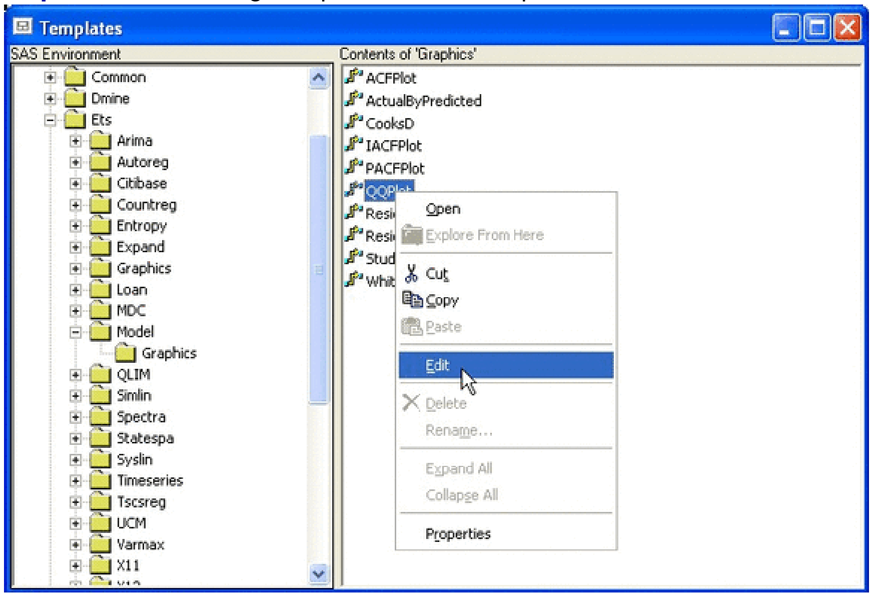
To display the default template definition, open the Templates window by typing **odstemplates** (or **odst** for short) in the command line. Expand **Sashelp.Tmplmst** and click on the **Ets** folder, as shown in Output 6.7.3.

Output 6.7.3. The Templates Window



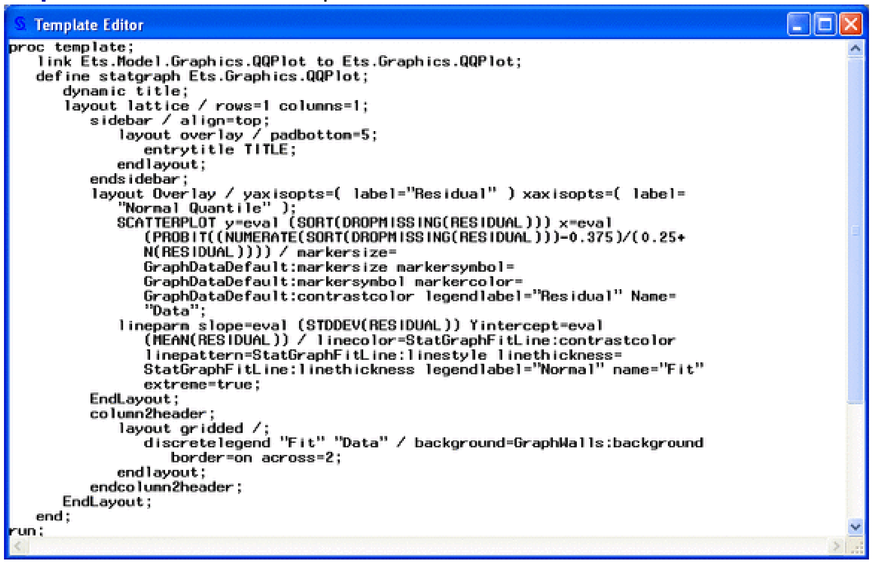
Next, open the **Model** folder and then open the **Graphics** folder. Then right-click on the “QQPlot” template icon and select **Edit**, as shown in Output 6.7.4.

Output 6.7.4. Editing Templates in the Template Window



Selecting **Edit** opens a Template Editor window, as shown in Output 6.7.5. You can use this window to edit the template.

Output 6.7.5. 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 6.7.5 is discussed below and in subsequent examples. It is listed in a more convenient format by the following statements:

```
proc template;
   source Ets.Model.Graphics.QQPlot;
   link Ets.Model.Graphics.QQPlot to Ets.Graphics.QQPlot;
   define statgraph Ets.Graphics.QQPlot;
      dynamic title;
      layout Overlay /
            yaxisopts=( label="Residual" )
```

```
                  xaxisopts=( label="Normal Quantile" );
               EntryTitle TITLE / pad=( bottom=5 );
               SCATTERPLOT
                  y=eval (SORT(DROPMISSING(RESIDUAL)))
                  x=eval (PROBIT((NUMERATE(SORT(DROPMISSING(RESIDUAL)))
                          -0.375)/(0.25 + N(RESIDUAL)))) /
                  markerattrs=
                     ( size=GraphDataDefault:markersize
                       symbol=GraphDataDefault:markersymbol
                       color=GraphDataDefault:contrastcolor )
                  legendlabel="Residual"
                  Name="Data";
               lineparm
                  slope=eval (STDDEV(RESIDUAL))
                  Y=eval (MEAN(RESIDUAL)) /
                  lineattrs=
                     ( color=GraphFitLine:contrastcolor
                       pattern=GraphFitLine:linepattern
                       thickness=GraphFitLine:linethickness )
                  legendlabel="Normal" name="Fit" extend=true;
               DiscreteLegend "Fit" "Data" /
                  across=2
                  border=on
                  backgroundattrs=( color=GraphWalls:color);
            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 169 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 6.7.1). 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.

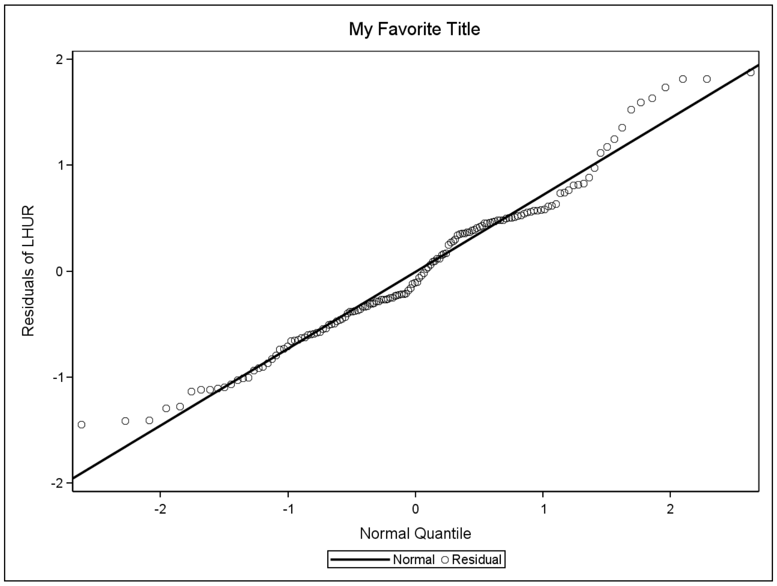
After submitting the PROC TEMPLATE statements you should see the following message in the SAS log:

```
  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 172.

Finally, resubmit the PROC MODEL statements on page 195 to display the Q-Q plot created with your modified template, as shown in Output 6.7.6.

Output 6.7.6. 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 171 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 6.6 to replay the Q-Q plot with the modified template.

See the "Reverting to Default Templates" section on page 171 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 6.8. Modifying Colors, Line Styles, and Markers

This example is a continuation of Example 6.7. 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 6.7.5, 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 MARKERATTRS= option of the SCATTERPLOT statement:

```
symbol = GraphDataDefault:markersymbol
```

The default value is a reference to the style attribute **`symbol`** of the style element **`GraphDataDefault`**. See the "Introducing Style Elements for Graphics" section on page 174 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 SYMBOL= option as follows.

```
symbol = 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.

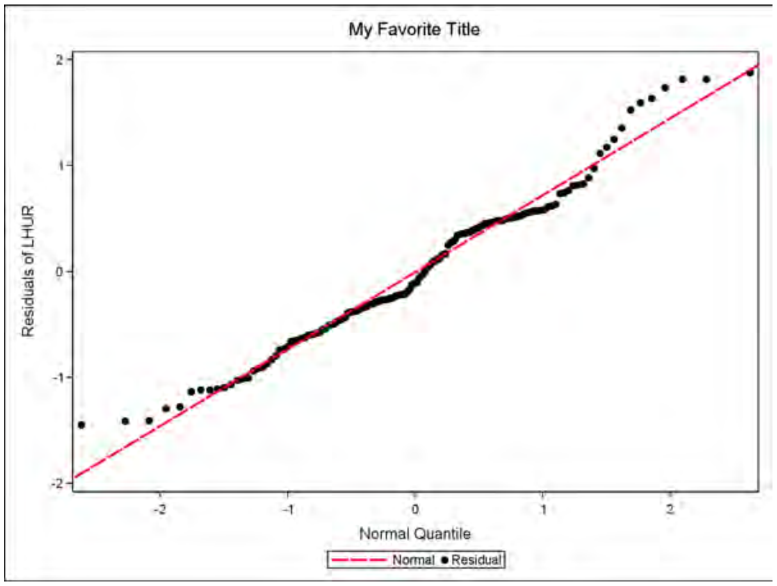
For more information about the syntax of the graphics template language and style elements for graphics, refer to the sections "TEMPLATE Procedure: Creating ODS Statistical Graphics Output (Experimental)" and "ODS Statistical Graphics and ODS Styles: Usage and Reference (Experimental)" at http://support.sas.com/documentation/onlinedoc/base/.

Similarly, you can change the line color and pattern with the COLOR= and PATTERN= options in the LINEATTRS= option of 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.

```
color   = red
pattern = dash
```

To display these modifications, shown in Output 6.8.1, submit the modified template definition and then resubmit the PROC MODEL statements on page 195. Alternatively, you can replay the plot using PROC DOCUMENT, as in Example 6.6.

Output 6.8.1. 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 6.9. 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 6.7 and Example 6.8, illustrates how to interchange the axes with a simple modification of the graph template.

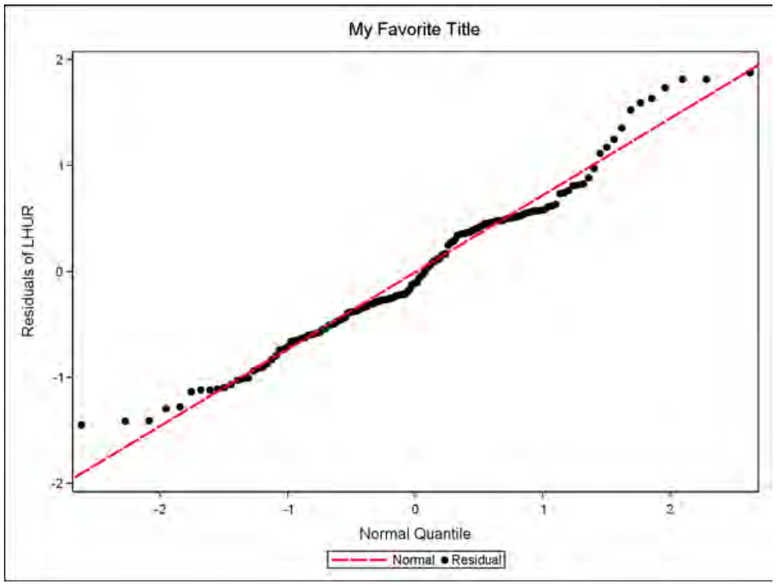
Begin by swapping the YAXISOPTS= and XAXISOPTS= options, and by swapping the X= and Y= options in the SCATTERPLOT statement.

Next, modify the LINEPARM statement. In Output 6.8.1, 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 6.7 and Example 6.8) is as follows:

```
proc template;
   source Ets.Model.Graphics.QQPlot;
   link Ets.Model.Graphics.QQPlot to Ets.Graphics.QQPlot;
   define statgraph Ets.Graphics.QQPlot;
      dynamic title;
      layout Overlay /
            xaxisopts=( label="Residuals of LHUR" )
            yaxisopts=( label="Normal Quantile" );
         EntryTitle "My Favorite Title" /
            pad=( bottom=5 );
         SCATTERPLOT
            x=eval (SORT(DROPMISSING(RESIDUAL)))
            y=eval (PROBIT((NUMERATE(SORT(DROPMISSING(RESIDUAL)))
                    -0.375)/(0.25 + N(RESIDUAL)))) /
            markerattrs=
               ( size=GraphDataDefault:markersize
                 symbol=CircleFilled
                 color=GraphDataDefault:contrastcolor )
            legendlabel="Residual"
            Name="Data";
         lineparm
            slope      = eval(1/STDDEV(RESIDUAL))
            Yintercept = eval(-MEAN(RESIDUAL)/STDDEV(RESIDUAL)) /
            lineattrs=
               ( color=red
                 pattern=dash
                 thickness=GraphFitLine:linethickness )
            legendlabel="Normal"
            name="Fit"
            extend=true;
         DiscreteLegend "Fit" "Data" /
            across=2
            border=on
            backgroundattrs=( color=GraphWalls:color)
          ;
      EndLayout;
   end;
run;
```

The resulting Q-Q plot, after submitting the preceding statements and the PROC MODEL statements on page 195, is shown in Output 6.9.1.

Output 6.9.1. 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 6.10. Modifying Tick Marks and Adding Grid Lines

This example, which is a continuation of Example 6.7, Example 6.8, and Example 6.9, illustrates how to modify the tick marks for an axis and suppress grid lines.

You can use the TICKVALUELIST= 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 6.9.1:

```
   yaxisopts = (label = "Normal Quantile"
                tickvaluelist = (-3 -2 -1 0 1 2))
```

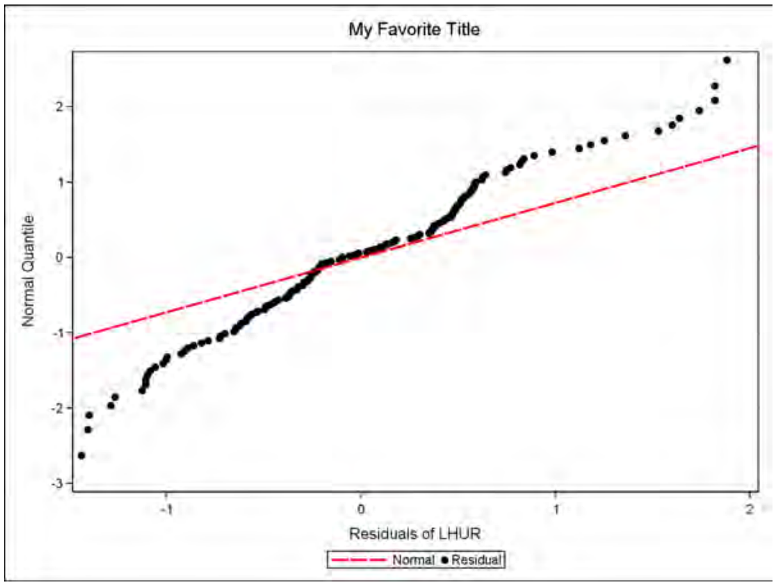
By default, the Q-Q plot in Output 6.9.1 does not display grid lines. You can request vertical grid lines only by specifying

```
   type=linear
```

in the YAXISOPTS= option of the LAYOUT statement.

The result of these changes, after submitting the modified template definition and the corresponding PROC MODEL statements on page 195, is displayed in Output 6.10.1.

Output 6.10.1. 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 6.11. 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 6.10.

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.

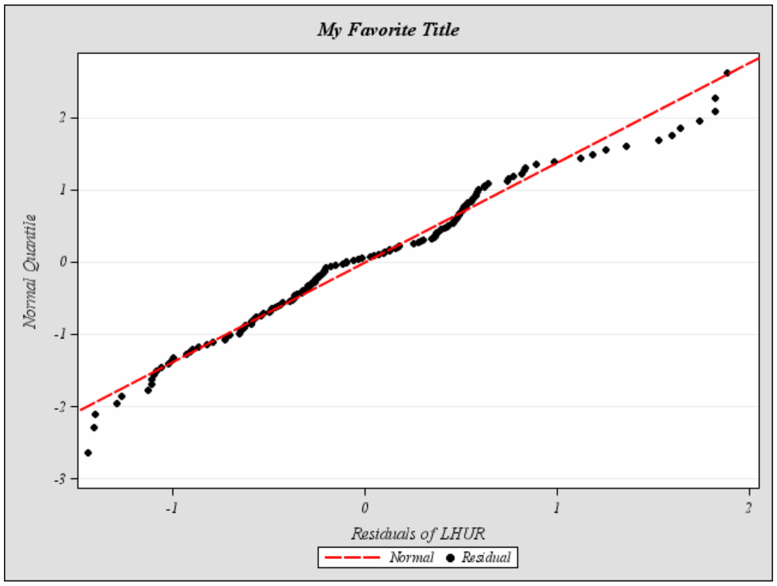
For more information about the DEFINE, PARENT, and REPLACE statements, refer to the “TEMPLATE Procedure: Creating a Style Definition” in the *SAS Output Delivery System User’s Guide*.

The Q-Q plots in the preceding examples, beginning with Example 6.6, were created with the “Default” style; see, for instance, Output 6.10.1. In contrast, the Q-Q plot displayed in Output 6.11.1 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 6.11.1. 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 175 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 6.12. Modifying Other Graph Elements in Styles

This example, which is a continuation of Example 6.11, 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 6.11.1 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 6.11.

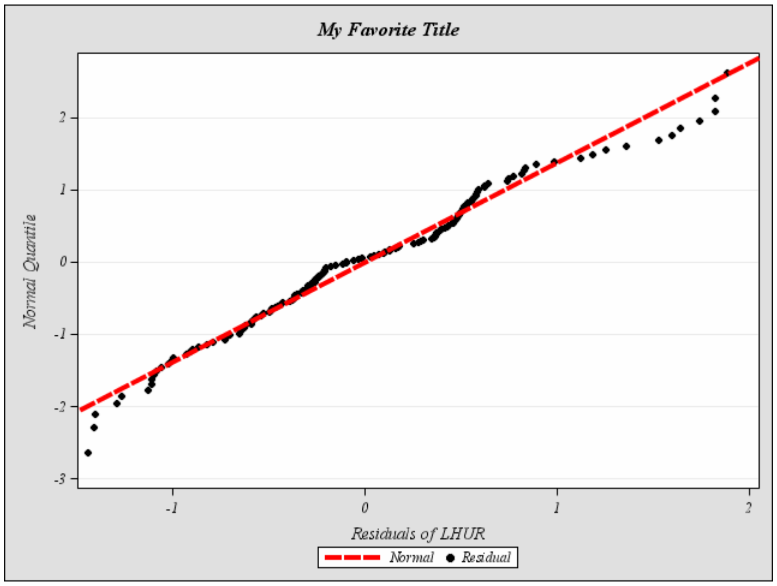
```
replace GraphFitLine /
   linethickness = 4px;
```

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);
   style GraphFit from GraphFit/
      linethickness = 4px;
end;
run;
```

Output 6.12.1 shows the Q-Q plot created by the MODEL statements on page 206 with the new version of NewStyle.

Output 6.12.1. 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 6.13. 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 6.12, 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;
```

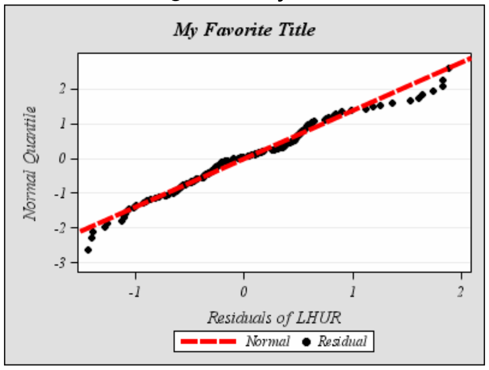
With all the changes introduced so far, NewStyle is defined as follows:

```
 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);
   style GraphFit from GraphFit/
      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 6.12.1, shown in Output 6.13.1, by specifying the preceding PROC TEMPLATE statements followed by the MODEL statements on page 206.

Output 6.13.1. 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 6.14. Modifying Panels

This example is taken from the “Getting Started” section of Chapter 66, “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 6.14.1 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 6.14.1.

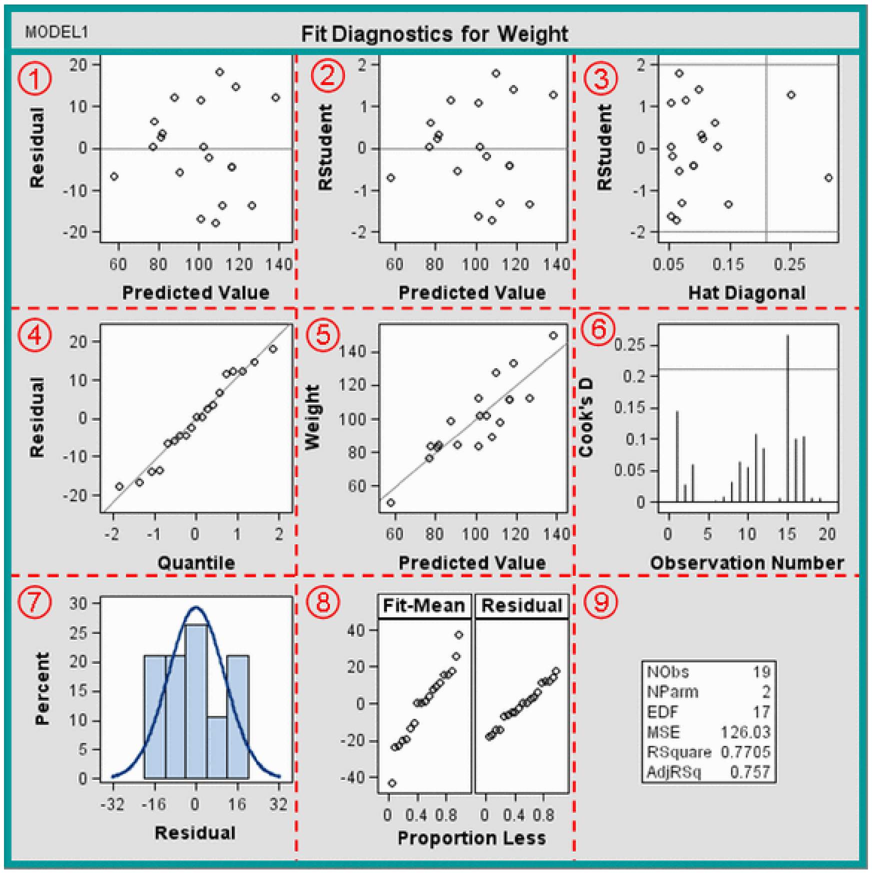
The following data are from a study of 19 children. The variables Height, Weight, and Age are measured for each child.

```
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 6.14.1.

```
ods html;
ods graphics on;
ods select DiagnosticsPanel;

proc reg data = Class;
   model Weight = Height;
run;
quit;

ods graphics off;
ods html close;
```

Output 6.14.1. 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 6.14.2. 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 168 for details on how to display the complete template definition).

```
proc template;
   source Stat.Reg.Graphics.DiagnosticsPanel;
   define statgraph Stat.Reg.Graphics.DiagnosticsPanel;
   notes "Diagnostics Panel";

/* Dynamic Variables */
dynamic _TITLE _MODELLABEL _DEPLABEL _NOBS _NPARM _EDF _MSE _RSquare _AdjRSq;

/* 3x3 LATTICE layout */
layout lattice / columns=3 rows=3 ... ;
   sidebar / align=top;
      layout overlay / ...
      endlayout;
   endsidebar;

/* 1. Residual by Predicted */
   layout overlay / ... ;
      lineparm slope=0 y=0 / extend ... ;
      scatterplot y=RESIDUAL x= ... ;
   endlayout;

...

/* LAYOUT statements for componenets 2-8 */

...

/* 9. Summary Statistics Box */
   layout overlay / pad=( left=5 right=5 );
      layout gridded / ... ;
         entry halign=left "NObs" / valign=top;
         entry halign=right eval (PUT(_NOBS,BEST6.)) / valign=top;
         .
         .
         .
         entry halign=left "AdjRSq" / valign=top;
         entry halign=right eval (PUT(_ADJRSQ,BEST6.)) / valign=top;
      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 6.14.1. 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 6.14.1) 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;
ine statgraph Stat.Reg.Graphics.DiagnosticsPanel;
notes "Diagnostics Panel";
dynamic _TITLE _MODELLABEL _DEPLABEL _NOBS _NPARM _EDF _MSE _RSquare _AdjRSq;

/* Begin 2x2 LATTICE layout */
layout lattice /
     columns   = 2
     rows      = 2 ...
   sidebar / align = top;
      layout overlay / pad=( bottom=5 );
         entrytitle halign       = left ...
      endlayout;
    endsidebar;

   /* 1. Residual By Predicted */
   layout overlay /
      yaxisopts = ...
      lineparm
         slope = 0
         y     = 0 /
         extend  ...
      scatterplot
         y = RESIDUAL
         x = PREDICTEDVALUE /
         markerattrs = ...
   endlayout;

   /* 4. Q-Q Plot */
   layout overlay /
      yaxisopts = ...
      lineparm
         slope  = eval(STDDEV(RESIDUAL))
         y      = ...
      scatterplot
         y = eval(SORT(DROPMISSING(RESIDUAL)))
         x = ...
   endlayout;

   /* Statements for components 6 and 7 (not listed) */

   /*  9. Summary Statistics Box in a SIDEBAR */
   sidebar / align = bottom;
      layout gridded;
         layout lattice /
             rows    = 1
             columns = 4
```

¶See page 169 for details on how to edit the template definition.

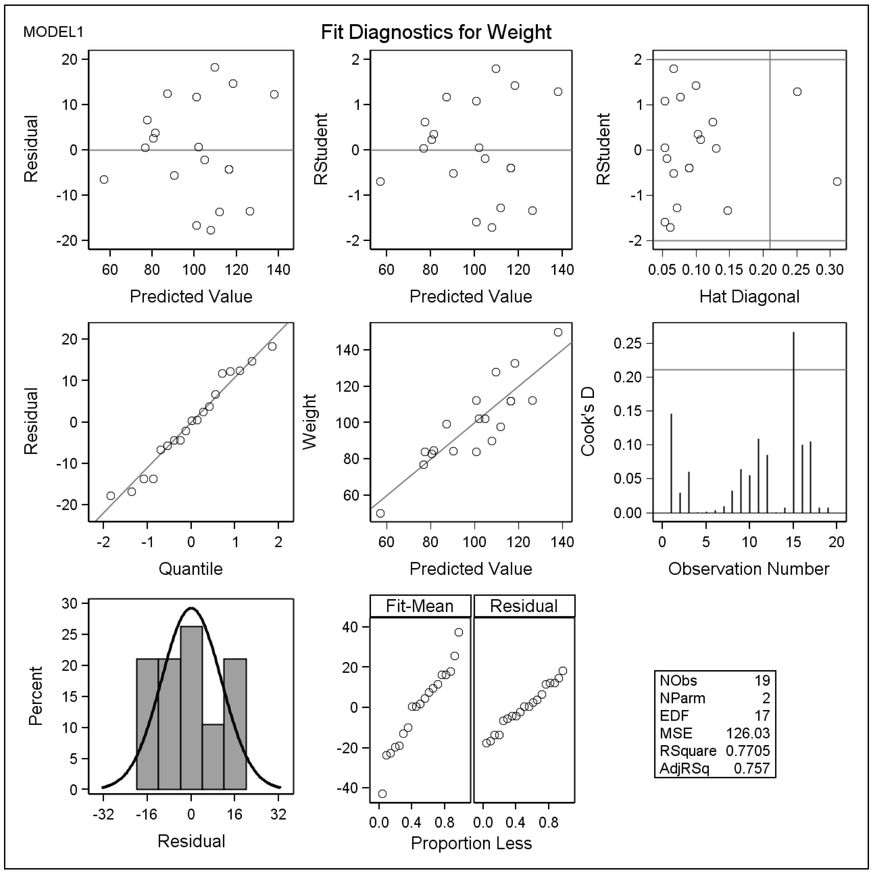
```
                 ...
            endlayout;
         endlayout;
      endsidebar;

      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. 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 6.14.2.

```
   ods html;
   ods graphics on;

   ods select DiagnosticsPanel;

   proc reg data = Class;
      model Weight = Height;
   run;
   quit;

   ods graphics off;
   ods html close;
```

Output 6.14.2. Simplified Diagnostics Panel



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.

Part 2
Procedure Reference

Contents

Chapter 7

The HPF Procedure

Chapter Contents

Chapter 7
The HPF Procedure

Overview

The HPF (High-Performance Forecasting) procedure provides a quick and automatic way to generate forecasts for many time series or transactional data in one step. The procedure can forecast millions of series at a time, with the series organized into separate variables or across BY groups.

- For typical time series, you can use the following smoothing models:
  - Simple
  - Double
  - Linear
  - Damped Trend
  - Seasonal
  - Winters Method (additive and multiplicative)
- Additionally, transformed versions of these models are provided:
  - Log
  - Square Root
  - Logistic
  - Box-Cox
- For intermittent time series (series where a large number of values are zero-valued), you can use an intermittent demand model such as Croston's Method and Average Demand Model.

Experimental graphics are now available with the HPF procedure. For more information, see the "ODS Graphics" section on page 257.

All parameters associated with the forecast model are optimized based on the data. Optionally, the HPF procedure can select the appropriate smoothing model for you using holdout sample analysis based on one of several model selection criteria.

The HPF procedure writes the time series extrapolated by the forecasts, the series summary statistics, the forecasts and confidence limits, the parameter estimates, and the fit statistics to output data sets. The HPF procedure optionally produces printed output for these results utilizing the Output Delivery System (ODS).

The HPF procedure can forecast both time series data, whose observations are equally spaced by a specific time interval (e.g., monthly, weekly), or transactional data, whose observations are not spaced with respect to any particular time interval. Internet, inventory, sales, and similar data are typical examples of transactional data. For transactional data, the data is accumulated based on a specified time interval to form a time series. The HPF procedure can also perform trend and seasonal analysis on transactional data.

Additionally, the Time Series Forecasting System of SAS/ETS software can be used to interactively develop forecasting models, estimate the model parameters, evaluate the models' ability to forecast, and display these results graphically. Refer to Chapter 34, "Overview of the Time Series Forecasting System," (*SAS/ETS User's Guide*) for details.

Also, the EXPAND procedure can be used for the frequency conversion and transformations of time series. Refer to Chapter 17, "The EXPAND Procedure," (*SAS/ETS User's Guide*) for details.

Getting Started

The HPF procedure is simple to use for someone who is new to forecasting, and yet at the same time it is powerful for the experienced professional forecaster who needs to generate a large number of forecasts automatically. 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 260.

Given an input data set that contains numerous time series variables recorded at a specific frequency, the HPF procedure can automatically forecast the series as follows:

```
PROC HPF DATA=<input-data-set> OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>;
   FORECAST <time-series-variables>;
   RUN;
```

For example, suppose that the input data set SALES contains numerous sales data recorded monthly, the variable that represents time is DATE, and the forecasts are to be recorded in the output data set NEXTYEAR. The HPF procedure could be used as follows:

```
proc hpf data=sales out=nextyear;
   id date interval=month;
   forecast _ALL_;
run;
```

The above statements automatically select the best fitting model, generate forecasts for every numeric variable in the input data set (SALES) for the next twelve months, and store these forecasts in the output data set (NEXTYEAR). Other output data sets can be specified to store the parameter estimates, forecasts, statistics of fit, and summary data.

If you want to print the forecasts using the Output Delivery System (ODS), then you need to add PRINT=FORECASTS:

```
 proc hpf data=sales out=nextyear print=forecasts;
    id date interval=month;
    forecast _ALL_;
 run;
```

Other results can be specified to output the parameter estimates, forecasts, statistics of fit, and summary data using ODS.

The HPF procedure can forecast both time series data, whose observations are equally spaced by a specific time interval (e.g., monthly, weekly), or transactional data, whose observations are not spaced with respect to any particular time interval.

Given an input data set containing transactional variables not recorded at any specific frequency, the HPF procedure accumulates the data to a specific time interval and forecasts the accumulated series as follows:

```
PROC HPF DATA=<input-data-set> OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
      ACCUMULATE=<accumulation>;
   FORECAST <time-series-variables>;
RUN;
```

For example, suppose that the input data set WEBSITES contains three variables (BOATS, CARS, PLANES), that are Internet data recorded on no particular time interval, and the variable that represents time is TIME, which records the time of the Web hit. The forecasts for the total daily values are to be recorded in the output data set NEXTWEEK. The HPF procedure could be used as follows:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

The above statements accumulate the data into a daily time series and automatically generate forecasts for the BOATS, CARS, and PLANES variables in the input data set (WEBSITES) for the next seven days and store the forecasts in the output data set (NEXTWEEK).

The HPF procedure can specify a particular forecast model or select from several candidate models based on a selection criterion. The HPF procedure also supports transformed models and holdout sample analysis.

Using the previous WEBSITES example, suppose that you want to forecast the BOATS variable using the best seasonal forecasting model that minimizes the mean absolute percent error (MAPE), the CARS variable using the best nonseasonal forecasting model that minimizes the mean square error (MSE) using holdout sample analysis on the last five days, and the PLANES variable using the Log Winters Method (additive). The HPF procedure could be used as follows:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats   / model=bests select=mape;
   forecast cars    / model=bestn select=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;
```

The above statements demonstrate how each variable in the input data set can be modeled differently and how several candidate models can be specified and selected based on holdout sample analysis or the entire range of data.

The HPF procedure is also useful in extending independent variables in (auto) regression models where future values of the independent variable are needed to predict the dependent variable.

Using the WEBSITES example, suppose that you want to forecast the ENGINES variable using the BOATS, CARS, and PLANES variable as regressor variables. Since future values of the BOATS, CARS, and PLANES are needed, the HPF procedure can be used to extend these variables in the future:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast engines / model=none;
   forecast boats   / model=bests select=mape;
   forecast cars    / model=bestn select=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;

proc autoreg data= nextweek;
   model engines = boats cars planes;
   output out=enginehits p=predicted;
run;
```

The above HPF procedure statements generate forecasts for BOATS, CARS, and PLANES in the input data set (WEBSITES) for the next seven days and extend the variable ENGINES with missing values. The output data set (NEXTWEEK) of the PROC HPF statement is used as an input data set for the PROC AUTOREG statement. The output data set of PROC AUTOREG contains the forecast of the variable ENGINE based on the regression model with the variables BOATS, CARS, and PLANES as regressors. See the AUTOREG procedure for details on autoregression.

The HPF procedure can also forecast intermittent time series (series where a large number of values are zero-valued). Typical time series forecasting techniques are less effective in forecasting intermittent time series.

For example, suppose that the input data set INVENTORY contains three variables (TIRES, HUBCAPS, LUGBOLTS) that are demand data recorded on no particular time interval, the variable that represents time is DATE, and the forecasts for the total weekly values are to be recorded in the output data set NEXTMONTH. The models requested are intermittent demand models, which can be specified as MODEL=IDM option. Two intermittent demand models are compared, Croston Model and Average Demand Model. The HPF procedure could be used as follows:

```
proc hpf data=inventory out=nextmonth lead=4 print=forecasts;
   id date interval=week accumulate=total;
   forecast tires hubcaps lugbolts / model=idm;
run;
```

In the above example, the total demand for inventory items is accumulated on a weekly basis and forecasts are generated that recommend future stocking levels.

Syntax

The following statements are used with the HPF procedure.

PROC HPF *options*;
  BY *variables*;
  FORECAST *variable-list / options*;
  ID *variable* **INTERVAL=** *interval options*;
  IDM *options*;

Functional Summary

The statements and options controlling the HPF procedure are summarized in the following table.

Description	**Statement**	**Option**
Statements		
specify BY-group processing	BY	
specify variables to forecast	FORECAST	
specify the time ID variable	ID	
specify intermittent demand model	IDM	
Data Set Options		
specify the input data set	PROC HPF	DATA=
specify to output forecasts only	PROC HPF	NOOUTALL
specify the output data set	PROC HPF	OUT=
specify parameter output data set	PROC HPF	OUTEST=
specify forecast output data set	PROC HPF	OUTFOR=
specify seasonal statistics output data set	PROC HPF	OUTSEASON=
specify statistics output data set	PROC HPF	OUTSTAT=
specify summary output data set	PROC HPF	OUTSUM=
specify trend statistics output data set	PROC HPF	OUTTREND=
replace actual values held back	FORECAST	REPLACEBACK

Description	Statement	Option
replace missing values	FORECAST	REPLACEMISSING
use forecast value to append	FORECAST	USE=
Accumulation and Seasonality Options		
specify accumulation frequency	ID	INTERVAL=
specify length of seasonal cycle	PROC HPF	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, FORECAST	ACCUMULATE=
specify missing value interpretation	ID, FORECAST	SETMISSING=
specify zero value interpretation	ID, FORECAST	ZEROMISS=
Forecasting Horizon, Holdout, Holdback Options		
specify data to hold back	PROC HPF	BACK=
specify forecast holdout sample size	FORECAST	HOLDOUT=
specify forecast holdout sample percent	FORECAST	HOLDOUTPCT=
specify forecast horizon or lead	PROC HPF	LEAD=
specify horizon to start summation	PROC HPF	STARTSUM=
Forecasting Model and Selection Options		
specify confidence limit width	FORECAST	ALPHA=
specify intermittency	FORECAST	INTERMITTENT=
specify forecast model	FORECAST	MODEL=
specify median forecats	FORECAST	MEDIAN
specify backcast initialization	FORECAST	NBACKCAST=
specify seasonality test	FORECAST	SEASONTEST=
specify model selection criterion	FORECAST	SELECT=
specify model transformation	FORECAST	TRANSFORM=
Intermittent Demand Model Options		
specify model for average demand	IDM	AVERAGE=
specify the base value	IDM	BASE=
specify model for demand intervals	IDM	INTERVAL=
specify model for demand sizes	IDM	SIZE=
Printing and Plotting Control Options		
specify graphical output	PROC HPF	PLOT=
specify printed output	PROC HPF	PRINT=
specify detailed printed output	PROC HPF	PRINTDETAILS

Description	Statement	Option
Miscellaneous Options		
specify that analysis variables are processed in sorted order	PROC HPF	SORTNAMES
limits error and warning messages	PROC HPF	MAXERROR=

PROC HPF Statement

PROC HPF *options;*

The following options can be used in the PROC HPF statement.

BACK= *n*
specifies the number of observations before the end of the data that the multistep forecasts are to begin. The default is BACK=0.

DATA= *SAS-data-set*
names the SAS data set containing the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

LEAD= *n*
specifies the number of periods ahead to forecast (forecast lead or horizon). The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation 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.

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.

NOOUTALL
specifies that only forecasts are written to the OUT= and OUTFOR= data sets. The NOOUTALL option includes only the final forecast observations in the output data sets, not the one-step forecasts for the data before the forecast period.

The OUT= and OUTFOR= data set will only contain the forecast results starting at the next period following the last observation to the forecast horizon specified by the LEAD= option.

OUT= *SAS-data-set*
names the output data set to contain the forecasts of the variables specified in the subsequent FORECAST statements. If an ID variable is specified, it will also be included in the OUT= data set. The values are accumulated based on the ACCUMULATE=

option and forecasts are appended to these values based on the FORECAST statement USE= option. The OUT= data set is particularly useful in extending the independent variables when forecasting dependent series associated with (auto) regression models. If the OUT= option is not specified, a default output data set DATAn is created. If you do not want the OUT= data set created, then use OUT=_NULL_.

OUTEST= *SAS-data-set*
names the output data set to contain the model parameter estimates and the associated test statistics and probability values. The OUTEST= data set is particularly useful for evaluating the significance of the model parameters and understanding the model dynamics.

OUTFOR= *SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, prediction standard error). The OUTFOR= data set is particularly useful for displaying the forecasts in tabular or graphical form.

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 ID statement INTERVAL= option or the SEASONALITY= option. The OUTSEASON= data set is particularly useful when analyzing transactional data for seasonal variations.

OUTSTAT= *SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is particularly useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series regardless of whether the HOLDOUT= option is specified.

OUTSUM= *SAS-data-set*
names the output data set to contain the summary statistics and the forecast summation. The summary statistics are based on the accumulated time series when the ACCUMULATE= or SETMISSING= options are specified. The forecast summations are based on the LEAD=, STARTSUM=, and USE= options. The OUTSUM= data set is particularly useful when forecasting 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 ID statement 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 HPF procedure produces no printed output. The following printing options are available:

ESTIMATES	prints the results of parameter estimation. (OUTEST= data set)
FORECASTS	prints the forecasts. (OUTFOR= data set)

PERFORMANCE	prints the performance statistics for each forecast.
PERFORMANCESUMMARY	prints the performance summary for each BY group.
PERFORMANCEOVERALL	prints the performance summary for all of the BY groups.
SEASONS	prints the seasonal statistics. (OUTSEASON= data set)
STATISTICS	prints the statistics of fit. (OUTSTAT= data set)
STATES	prints the backcast, initial, and final states.
SUMMARY	prints the summary statistics for the accumulated time series. (OUTSUM= data set)
TRENDS	prints the trend statistics. (OUTTREND= data set)
ALL	Same as PRINT=(ESTIMATES FORECASTS STATISTICS SUMMARY). PRINT=(ALL,TRENDS,SEASONS) prints all of the options listed above.

For example, PRINT=FORECASTS prints the forecasts, PRINT=(ESTIMATES, FORECASTS) prints the parameter estimates and the forecasts, and PRINT=ALL prints all of the above output.

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 observations forms a seasonal cycle. The SEASONALITY= option is applicable only for seasonal forecasting models. 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 FORECAST statements are processed in sorted order.

STARTSUM= *n*

specifies the starting forecast lead (or horizon) for which to begin summation of the forecasts specified by the LEAD= option. The STARTSUM= value must be less than the LEAD= value. The default is STARTSUM=1, that is, the sum from the one-step ahead forecast to the multistep forecast specified by the LEAD= option.

The prediction standard errors of the summation of forecasts takes into account the correlation between the multistep forecasts. The DETAILS section describes the STARTSUM= option in more detail.

BY Statement

BY *variables;*

A BY statement can be used with PROC HPF to obtain separate analyses for groups of observations defined by the BY variables.

FORECAST Statement

FORECAST *variable-list / options*;

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast. The options specify which forecast model is to be used or how the forecast model is selected from several possible candidate models.

A data set variable can be specified in only one FORECAST statement. Any number of FORECAST statements can be used. The following options can be used with the FORECAST statement.

ACCUMULATE= *option*

specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

ALPHA= *number*

specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1. The default is ALPHA=0.05, which produces 95% confidence intervals.

HOLDOUT= *n*

specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series ending at the last nonmissing observation. If the ACCUMULATE= option is specified, the holdout sample is based on the accumulated series. If the holdout sample is not specified, the full range of the actual time series is used for model selection.

For each candidate model specified, the holdout sample is excluded from the initial model fit and forecasts are made within the holdout sample time range. Then, for each candidate model specified, the statistic of fit specified by the SELECT= option is computed using only the observations in the holdout sample. Finally, the candidate model, which performs best in the holdout sample, based on this statistic, is selected to forecast the actual time series.

The HOLDOUT= option is only used to select the best forecasting model from a list of candidate models. After the best model is selected, the full range of the actual time series is used for subsequent model fitting and forecasting. It is possible that one model will outperform another model in the holdout sample but perform less well when the entire range of the actual series is used.

If MODEL=BESTALL and HOLDOUT= options are used together, the last one hundred observations are used to determine whether the series is intermittent. If the series determined not to be intermittent, holdout sample analysis will be used to select the smoothing model.

HOLDOUTPCT= *number*

specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where T is the length of the time series with beginning and ending missing values removed. The default is 100 (100%).

INTERMITTENT= *number*

specifies a number greater than one which is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number then the series is assumed to be intermittent. This option is used with MODEL=BESTALL option. The default is INTERMITTENT=1.25.

MEDIAN

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series using the TRANSFORM= option, the mean and median forecast values are identical.

MODEL= *model-name*

specifies the forecasting model to be used to forecast the actual time series. A single model can be specified or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the SELECT= option and the HOLDOUT= option. The default is MODEL=BEST. The following forecasting models are provided:

NONE	No forecast. The accumulated time series is appended with missing values in the OUT= data set. This option is particularly useful when the results stored in the OUT= data set are subsequently used in (auto) regression analysis where forecasts of the independent variables are needed to forecast the dependent variable.
SIMPLE	Simple (Single) Exponential Smoothing
DOUBLE	Double (Brown) Exponential Smoothing
LINEAR	Linear (Holt) Exponential Smoothing
DAMPTREND	Damped Trend Exponential Smoothing
SEASONAL	Seasonal Exponential Smoothing
WINTERS	Winters Multiplicative Method
ADDWINTERS	Winters Additive Method
BEST	Best Candidate Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND, SEASONAL, WINTERS, ADDWINTERS)
BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)

BESTS	Best Candidate Seasonal Smoothing Model (SEASONAL, WINTERS, ADDWINTERS)
IDM|CROSTON	Intermittent Demand Model such as Croston's Method or Average Demand Model. An intermittent time series is one whose values are mostly zero.
BESTALL	Best Candidate Model (IDM, BEST)

The BEST, BESTN, and BESTS options specify a group of models by which the HOLDOUT= option and SELECT= option are used to select the model used to forecast the accumulated time series based on holdout sample analysis. Transformed versions of the above smoothing models can be specified using the TRANSFORM= option.

The BESTALL option specifies that if the series is intermittent, an intermittent demand model such as Croston's Method or Average Demand Model (MODEL=IDM) is selected; otherwise; the best smoothing model is selected (MODEL=BEST). Intermittency is determined by the INTERMITTENT= option.

The documentation for Chapter 19, "Forecasting Process Details," describes the above smoothing models and intermittent models in greater detail.

NBACKCAST= *n*

specifies the number of observations used to initialize the backcast states. The default is the entire series.

REPLACEBACK

specifies that actual values excluded by the BACK= option are replaced with one-step-ahead forecasts in the OUT= data set.

REPLACEMISSING

specifies that embedded missing actual values are replaced with one-step-ahead forecasts in the OUT= data set.

SEASONTEST= *option*

specifies the options related to the seasonality test. This option is used with MODEL=BEST and MODEL=BESTALL option.

The following options are provided:

SEASONTEST=NONE	no test
SEASONTEST=(SIGLEVEL=number)	Significance probability value.

Series with strong seasonality have small test probabilities. SEASONTEST=(SIGLEVEL=0) always implies seasonality. SEASONTEST=(SIGLEVEL=1) always implies no seasonality. The default is SEASONTEST=(SIGLEVEL=0.01).

SELECT= *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option would often be used in conjunction with the

HOLDOUT= option. The default is SELECT=RMSE. The following statistics of fit are provided:

SSE	Sum of Square Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
UMSE	Unbiased Mean Square Error
URMSE	Unbiased Root Mean Square Error
MAXPE	Maximum Percent Error
MINPE	Minimum Percent Error
MPE	Mean Percent Error
MAPE	Mean Absolute Percent Error
MDAPE	Median Percent Error
GMAPE	Geometric Mean Percent Error
MINPPE	Minimum Predictive Percent Error
MAXPPE	Maximum Predictive Percent Error
MPPE	Mean Predictive Percent Error
MAPPE	Symmetric Mean Absolute Predictive Percent Error
MDAPPE	Median Predictive Percent Error
GMAPPE	Geometric Mean Predictive Percent Error
MINSPE	Minimum Symmetric Percent Error
MAXSPE	Maximum Symmetric Percent Error
MSPE	Mean Symmetric Percent Error
SMAPE	Symmetric Mean Absolute Percent Error
MDASPE	Median Symmetric Percent Error
GMASPE	Geometric Mean Symmetric Percent Error
MINRE	Minimum Relative Error
MAXRE	Maximum Relative Error
MRE	Mean Relative Error
MRAE	Mean Relative Absolute Error
MDRAE	Median Relative Absolute Error
GMRAE	Geometric Mean Relative Absolute Error
MAXERR	Maximum Error
MINERR	Minimum Error
ME	Mean Error
MAE	Mean Absolute Error
RSQUARE	R-Square

ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion

SETMISSING= *option* **|** *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST 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 actual 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
AUTO	Automatically choose between NONE and LOG based on model selection criteria.

When the TRANSFORM= option is specified the time series must be strictly positive. Once the time series is transformed, the model parameters are estimated using the transformed series. The forecasts of the transformed series are then computed, and finally, the transformed series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

The TRANSFORM= option is not applicable when MODEL=IDM is specified.

USE= option

specifies which forecast values are appended to the actual values in the OUT= and OUTSUM= data sets. The following USE= options are provided:

PREDICT	The predicted values are appended to the actual values. This option is the default.
LOWER	The lower confidence limit values are appended to the actual values.

UPPER	The upper confidence limit values are appended to the actual values.

Thus, the USE= option enables the OUT= and OUTSUM= data sets to be used for worst/best/average/median case decisions.

ZEROMISS= *option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

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 actual time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the actual time series. The information specified affects all variables specified in subsequent FORECAST 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 actual time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations coinciding with a particular time period (e.g., 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 nonmissing 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 option and the START= option can be used to ensure that data associated with each BY group contains the same number of observations.

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. 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.

The basic intervals are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, DAY, HOUR, MINUTE, SECOND. Refer to SAS/ETS User's Guide chapter on Date Interval, Foremats, and Functions for the intervals that can be specified.

NOTSORTED

specifies that the time ID values are not in sorted order. The HPF 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 assigned 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 nonmissing value.
LAST	Missing values are set to the accumulated last nonmissing value.
PREVIOUS | PREV	Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

If SETMISSING=MISSING is specified and the MODEL= option specifies a smoothing model, the missing observations are smoothed over. If MODEL=IDM is specified, missing values are assumed to be periods of no demand, that is, SETMISSING=MISSING is equivalent to SETMISSING=0.

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 option and the END= option can be used to ensure that data associated with each by group contains the same number of observations.

ZEROMISS= *option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following options can also be used to determine how beginning and/or ending zero values are assigned:

NONE	Beginning and/or ending zeros unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing and/or zero the series is not changed.

IDM Statement

IDM *options;*

The IDM statement is used to specify an intermittent demand model. An intermittent demand series can be analyzed in two ways: individually modeling both demand interval and size component or jointly modeling these components using the average demand component (demand size divided by demand interval). The IDM statement specifies the two smoothing models to be used to forecast the demand interval component (INTERVAL= option) and the demand size component (SIZE= option), or specifies the single smoothing model to be used to forecast the average demand component (AVERAGE= option). Therefore, two smoothing models (INTERVAL= and SIZE= options) must be specified or one smoothing model (AVERAGE= option) must be specified. Only one statement can be specified.

The following examples illustrate typical uses of the IDM statement:

```
/* default specification */
idm;

/* demand interval model only specification */
idm interval=(transform=log);

/* demand size model only specification */
idm size=(method=linear);
```

```
/* Croston's Method */
idm interval=(method=simple)
    size    =(method=simple);

/* Log Croston's Method */
idm interval=(method=simple transform=log)
    size    =(method=simple transform=log);

/* average demand model specification */
idm average=(method=bestn);
```

The default specification uses both the INTERVAL= option and SIZE= option defaults for the decomposed (Croston's) demand model and the AVERAGE= option defaults for the average demand model.

The following example illustrates how to automatically choose the decomposed demand model using MAPE as the model selection criterion:

```
idm interval=(method=simple transform=auto select=mape)
    size    =(method=simple transform=auto select=mape);
forecast sales / model=idm select=mape;
```

The preceding fits two forecast models (simple and log simple exponential smoothing) to both the demand interval and size components. The forecast model that results in the lowest in-sample MAPE for each component is used to forecast the component.

The following example illustrates how to automatically choose the average demand model using MAPE as the model selection criterion:

```
idm average=(method=simple transform=auto select=mape);
forecast sales / model=idm;
```

The preceding fits two forecast models (simple and log simple exponential smoothing) to the average demand component. The forecast model that results in the lowest in-sample MAPE is used to forecast the component.

Combining the above two examples, the following example illustrates how to automatically choose between the decomposed demand model and the average demand model using MAPE as the model selection criterion:

```
idm interval=(method=simple transform=auto select=mape)
    size    =(method=simple transform=auto select=mape)
    average =(method=simple transform=auto select=mape);
forecast sales / model=idm select=mape;
```

The preceding automatically selects between the decomposed demand model and the average demand model as described previously. The forecast model that results in the lowest in-sample MAPE is used to forecast the series.

The following options can be specified in the IDM statement:

INTERVAL=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand interval component. See smoothing model specification options described below.

SIZE=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand size component. See smoothing model specification options described below.

AVERAGE=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand average component. See smoothing model specification options described below.

BASE=*AUTO* | *number*
specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's Method use BASE=0 which defines departures based on zero. The default is BASE=0.

Given a time series, y_t, and base value, b, the time series is adjusted by the base value to create the base adjusted time series, $x_t = y_t - b$. Demands are assumed to occur when the base adjusted series is nonzero (or when the time series, y_t, departs from the base value, b).

When BASE=AUTO, the base value is automatically determined by the time series median, minimum, and maximum value and the INTERMITTENT= option value of the FORECAST statement.

Smoothing Model Specification Options for IDM Statement

The smoothing model options describe how to forecast the demand interval, size, and average demand components (INTERVAL= option, SIZE= option, and AVERAGE= option).

If the smoothing model options are not specified, the following are the defaults for the demand interval, size, and average components.

```
interval=(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
          trendrest=(0.0001 0.9999)
          damprest =(0.0001 0.9999) select=rmse bounds=(1,.));

size    =(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
          trendrest=(0.0001 0.9999)
          damprest =(0.0001 0.9999) select=rmse);

average  =(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
```

```
            trendrest=(0.0001 0.9999)
            damprest =(0.0001 0.9999) select=rmse);
```

The above smoothing model options provide the typical automation in intermittent demand model selection.

The following describes the smoothing model options:

TRANSFORM= *option*

specifies the time series transformation to be applied to the demand component. The following transformations are provided:

NONE	No transformation is applied.
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
AUTO	Automatically choose between NONE and LOG based on model selection criteria. This option is the default.

When the TRANSFORM= option is specified, the demand component must be strictly positive. Once the demand component is transformed, the model parameters are estimated using the transformed component. The forecasts of the transformed component are then computed, and finally, the transformed component forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

MEDIAN

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series using the TRANSFORM= option, the mean and median component forecast values are identical.

METHOD= *method-name*

specifies the forecasting model to be used to forecast the demand component. A single model can be specified or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the SELECT= option of the IDM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

SIMPLE	Simple (Single) Exponential Smoothing
DOUBLE	Double (Brown) Exponential Smoothing
LINEAR	Linear (Holt) Exponential Smoothing
DAMPTREND	Damped Trend Exponential Smoothing

BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)

NOSTABLE
specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

LEVELPARM= *number*
specifies the level weight parameter initial value. See the smoothing model parameter specifications options below.

LEVELREST=(*number*,*number***)**
specifies the level weight parameter restrictions. See the smoothing model parameter specifications options below.

TRENDPARM= *number*
specifies the trend weight parameter initial value. See the smoothing model parameter specifications options below.

TRENDREST=(*number*,*number***)**
specifies the trend weight parameter restrictions. See the smoothing model parameter specifications options below.

DAMPPARM= *number*
specifies the damping weight parameter initial value. See the smoothing model parameter specifications options below.

DAMPREST=(*number*,*number***)**
specifies the damping weight parameter restrictions. See the smoothing model parameter specifications options below.

NOEST
specifies that the smoothing model parameters are fixed values. To use this option, all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

BOUNDS=(*number*,*number***)**
Specifies the component forecast bound. See the smoothing model forecast bounds below.

SELECT= *option*
specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option would often be used in conjunction with the HOLDOUT= option specified in the FORECAST statement. The default is SELECT=RMSE. The statistics of fit provided are the same as those provided in the FORECAST statement.

Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified the default is (0.0001 0.9999), which implies that the parameters are restricted between 0.0001 and 0.9999. Parameters and their restrictions are required to be greater than or equal to -1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

Smoothing Model Forecast Bounds Options

Specifies the demand component forecast bounds. The forecast bounds restrict the component forecasts. The default for demand interval forecasts is BOUNDS=1. The lower bound for the demand interval forecast must be greater than one. The default for demand size forecasts is BOUNDS=(.,.) or no bounds. The demand size forecasts bounds are applied after the forecast is adjusted by the base value.

Details

The HPF 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, suppose the transactional data ranges from June 1990 to January 2000, 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 HPF procedure can be used to forecast time series data as well as transactional data. If the data is transactional, then the procedure must first accumulate the data into a time series before it can be forecast. The procedure uses the following sequential steps to produce forecasts, with the options that control the step listed to the right:

1. Accumulation	ACCUMULATE= option
2. Missing Value Interpretation	SETMISSING= option
3. Diagnostic Tests	INTERMITTENT= and SEASONTEST= options
4. Model Selection	MODEL=, HOLDOUT=, HOLDOUTPCT=, and SELECT= options
5. Transformations	TRANSFORM= option
6. Parameter Estimation	MODEL= option
7. Forecasting	MODEL= and LEAD= options
8. Inverse Transformation	TRANSFORM= and MEDIAN options
9. Statistics of Fit	SELECT= option

10. Summation of Forecasts	LEAD= and STARTSUM= options

Each of the above steps is described below.

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 actual 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:

```
O1MAR1999    14.14
O1APR1999     .
O1MAY1999    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. The forecasting models used by the HPF procedure can effectively handle missing values (see the "Missing Value Modeling Issues" section on page 248). But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no (zero) value. 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.

Diagnostic Tests

The INTERMITTENT= option set the thresholds for categorizing a series as intermittent or non-intermittent. The SEASONTEST= option set the thresholds for categorizing a series as seasonal or non-seasonal.

Model Selection

When more than one candidate model is specified, forecasts for each candidate model are compared using the model selection criterion specified by the SELECT= option. The selection criterion is computed using the multistep forecasts in the holdout sample range if the HOLDOUT= or HOLDOUTPCT= options are specified, or the one-step ahead forecasts for the full range of the time series if the HOLDOUT= and HOLDOUTPCT= options are not specified. The candidate model with the best selection criterion is selected to forecast the time series.

Transformations

If the TRANSFORM= option is specified, the time series is transformed prior to model parameter estimation and forecasting. Only strictly positive series can be transformed. An error is generated when the TRANSFORM= option is used with a nonpositive series.

Parameter Estimation

All parameters associated with the model are optimized based on the data with the default parameter restrictions imposed. If the TRANSFORM= option is specified, the transformed time series data are used to estimate the model parameters.

Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. For the smoothing models, 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. See Chapter 19, "Forecasting Process Details," for greater detail on how missing values are treated in the smoothing models. For MODEL=IDM, specified missing values are assumed to be periods of no demand.

The treatment of missing values can also be specified by the user with the SETMISSING= option, which changes the missing values prior to modeling.

Even though all of the observed data are nonmissing, using the ACCUMULATE= option can create missing values in the accumulated series.

Forecasting

Once the model parameters are estimated, one-step ahead forecasts are generated for the full range of the actual (optionally transformed) time series data, and multistep forecasts are generated from the end of the observed time series to the future time period after the last observation specified by the LEAD= option. If there are missing values at the end of the time series, the forecast horizon will be greater than that specified by the LEAD= option.

Inverse Transformations

If the TRANSFORM= option is specified, the forecasts of the transformed time series are inverse transformed. By default, the mean (expected) forecasts are generated. If the MEDIAN option is specified, the median forecasts are generated.

Statistics of Fit

The statistics of fit (or goodness-of-fit statistics) are computed by comparing the actual time series data and the generated forecasts. If the TRANSFORM= option was specified, the statistics of fit are based on the inverse transformed forecasts.

Forecast Summation

The multistep forecasts generated by the above steps are summed from the STARTSUM= number to the LEAD= number. For example, if STARTSUM=4 and LEAD=6, the 4-step through 6-step ahead forecasts are summed. The predictions are simply summed. However, the prediction error variance of this sum is computed taking into account the correlation between the individual predictions. The upper and lower confidence limits for the sum of the predictions is then computed based on the prediction error variance of the sum.

The forecast summation is particularly useful when it is desirable to model in one frequency yet the forecast of interest is another frequency. For example, if a time series has a monthly frequency (INTERVAL=MONTH) and you want a forecast for the third and fourth future months, a forecast summation for the third and fourth month can be obtained by specifying STARTSUM=3 and LEAD=4.

Variance-related computations are only computed when no transformation is specified (TRANSFORM=NONE).

Comparison to the Time Series Forecasting System

With the exception of Model Selection, the techniques used in the HPF procedure are identical to the Time Series Forecasting System of SAS/ETS software. For Model Parameter Estimation, the default parameter restrictions are imposed.

Data Set Output

The HPF procedure can create the OUT=, OUTEST=, OUTFOR=, OUTSTAT=, OUTSUM=, OUTSEASON=, and OUTTREND= data sets. In general, these data sets will contain the variables listed in the BY statement. In general, if a forecasting 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 FORECAST 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 FORECAST statements are accumulated based on the ACCUMULATE= option and missing values are interpreted based on the SETMISSING= option. If the REPLACEMISSING option is specified, missing values embedded missing values are replaced by the one step-ahead forecasts.

These variable values are then extrapolated based on their forecasts or extended with missing values when the MODEL=NONE option is specified. If USE=LOWER is specified, the variable is extrapolated with the lower confidence limits; if USE=UPPER, the variable is extrapolated using the upper confidence limits; otherwise, the variable values are extrapolated with the predicted values. If the TRANSFORM= option is specified, the predicted values will contain either mean or median forecasts depending on whether or not the MEDIAN option is specified.

If any of the forecasting steps fail for particular variable, the variable values are extended by missing values.

OUTEST= Data Set

The OUTEST= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the parameter estimation step:

NAME	Variable name
MODEL	Forecasting Model
TRANSFORM	Transformation
PARM	Parameter Name
EST	Parameter Estimate
STDERR	Standard Errors
TVALUE	*t*-Values
PVALUE	Probability Values

If the parameter estimation step fails for a particular variable, no observations are recorded.

OUTFOR= Data Set

The OUTFOR= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the forecasting step:

NAME	Variable name
TIMEID	Time ID values
PREDICT	Predicted Values
STD	Prediction Standard Errors
LOWER	Lower Confidence Limits
UPPER	Upper Confidence Limits
ERROR	Prediction Errors

If the forecasting step fails for a particular variable, no observations are recorded. If the TRANSFORM= option is specified, the values in the variables listed above are the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are stored; otherwise, the mean forecasts are stored.

OUTSTAT= Data Set

The OUTSTAT= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables contain observations related to the statistics of fit step:

NAME	Variable name
REGION	Statistics of Fit Region
DFE	Degrees of Freedom Error
N	Number of Observations
NOBS	Number of Observations Used
NMISSA	Number of Missing Actuals
NMISSP	Number of Missing Predicted Values
NPARMS	Number of parameters
SSE	Sum of Square Error

MSE	Mean Square Error
UMSE	Unbiased Mean Square Error
RMSE	Root Mean Square Error
URMSE	Unbiased Root Mean Square Error
MAPE	Mean Absolute Percent Error
MAE	Mean Absolute Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion
MAXERR	Maximum Error
MINERR	Minimum Error
MINPE	Maximum Percent Error
MAXPE	Minimum Percent Error
ME	Mean Error
MPE	Mean Percent Error

If the statistics of fit step fails for particular variable, no observations are recorded. If the TRANSFORM= option is specified, the values in the variables listed above are computed based on the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are the basis; otherwise, the mean forecasts are the basis.

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 summary statistics for each variable specified in a FORECAST statement. For variables listed in FORECAST statements where the option MODEL=NONE is specified, the values related to forecasts are set to missing. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the forecast values are set based on the USE= option.

Variables related to summary statistics are based on the ACCUMULATE= and SETMISSING= options:

NAME	Variable name
STATUS	Forecasting Status. Nonzero values imply that no forecast was generated for the series.

NOBS	Number of Observations
N	Number of Nonmissing Observations
NMISS	Number of Missing Observations
MIN	Minimum Value
MAX	Maximum Value
MEAN	Mean Value
STDDEV	Standard Deviation

Variables related to forecast summation are based on the LEAD= and STARTSUM= options:

PREDICT	Forecast Summation Predicted Values
STD	Forecast Summation Prediction Standard Errors
LOWER	Forecast Summation Lower Confidence Limits
UPPER	Forecast Summation Upper Confidence Limits

Variance-related computations are only computed when no transformation is specified (TRANSFORM=NONE).

Variables related to multistep forecast based on the LEAD= and USE= options:

LEADn	Multistep Forecast (n ranges from one to LEAD=number). If USE=LOWER, this variable will contain the lower confidence limits; if USE=UPPER, this variable will contain the upper confidence limits; otherwise, this variable will contain the predicted values.

If the forecast step fails for a particular variable, the variables related to forecasting are set to missing. The OUTSUM= data set contains both a summary of the (accumulated) time series and optionally its forecasts for all series.

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 FORECAST statement.

Variables related to seasonal statistics are based on the INTERVAL= or SEASONALITY= options:

NAME	Variable name
TIMEID	Time ID values
SEASON	Seasonal index
NOBS	Number of Observations

N	Number of Nonmissing Observations
NMISS	Number of Missing Observations
MIN	Minimum Value
MAX	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

The above statistics are computed for each season.

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 FORECAST statement.

Variables related to trend statistics are based on the INTERVAL= and SEASONALITY= options:

NAME	Variable name
TIMEID	Time ID values
SEASON	Seasonal index
NOBS	Number of Observations
N	Number of Nonmissing Observations
NMISS	Number of Missing Observations
MIN	Minimum Value
MAX	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

The above statistics are computed for each time period.

Printed Output

The HPF 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 HPF statement. In general, if a forecasting 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=SUMMARY

prints the summary statistics and forecast summaries similar to the OUTSUM= data set.

PRINT=ESTIMATES

prints the parameter estimates similar to the OUTEST= data set.

PRINT=FORECASTS

prints the forecasts similar to the OUTFOR= data set. For MODEL=IDM, a table containing demand series is also printed.

If the MODEL=IDM option is specified, the demand series predictions table is also printed. This table is based on the demand index (when demands occurred).

PRINT=PERFORMANCE

prints the performance statistics.

PRINT=PERFORMANCESUMMARY

prints the performance summary for each BY group.

PRINT=PERFORMANCEOVERALL

prints the performance summary for all BY groups.

PRINT=STATES

prints the backcast, initial, and final smoothed states.

PRINT=SEASONS

prints the seasonal statistics similar to the OUTSEASON= data set.

PRINT=STATISTICS

prints the statistics of fit similar to the OUTSTAT= data set.

PRINT=TRENDS

Prints the trend statistics similar to the OUTTREND= data set.

PRINTDETAILS

The PRINTDETAILS option is the opposite of the NOOUTALL option.

Specifically, if PRINT=FORECASTS and the PRINTDETAILS options are specified, the one-step ahead forecasts, throughout the range of the data, are printed as well as the information related to a specific forecasting model such as the smoothing states. If the PRINTDETAILS option is not specified, only the multistep forecasts are printed.

ODS Table Names

The table below relates the PRINT= options to ODS tables:

Table 7.1. ODS Tables Produced in PROC HPF

ODS Table Name	Description	Option
ODS Tables Created by the PRINT=SUMMARY option		
DescStats	Descriptive Statistics	
DemandSummary	Demand Summary	MODEL=IDM option only
ForecastSummary	Forecast Summary	
ForecastSummmation	Forecast Summation	
ODS Tables Created by the PRINT=ESTIMATES option		
ModelSelection	Model Selection	
ParameterEstimates	Parameter Estimates	
ODS Tables Created by the PRINT=FORECASTS option		
Forecasts	Forecast	
Demands	Demands	MODEL=IDM option only
ODS Tables Created by the PRINT=PERFORMANCE option		
Performance	Performance Statistics	
ODS Tables Created by the PRINT=PERFORMANCESUMMARY option		
PerformanceSummary	Performance Summary	
ODS Tables Created by the PRINT=PERFORMANCEOVERALL option		
PerformanceSummary	Performance Overall	
ODS Tables Created by the PRINT=SEASONS option		
SeasonStatistics	Seasonal Statistics	
ODS Tables Created by the PRINT=STATES option		
SmoothedStates	Smoothed States	
DemandStates	Demand States	MODEL=IDM option only
ODS Tables Created by the PRINT=STATISTICS option		
FitStatistics	Statistics of Fit	
ODS Tables Created by the PRINT=TRENDS option		
TrendStatistics	Trend Statistics	

The ODS table ForecastSummary is related to all time series within a BY group. The other 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 HPF 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= option in the HPF statement according to the following syntax. For more information on the ODS GRAPHICS statement, refer to Chapter 9, "Statistical Graphics Using ODS" (*SAS/ETS User's Guide*).

PLOT= *option* **|** **(***options***)**

specifies the graphical output desired. By default, the HPF procedure produces no graphical output. The following printing options are available:

ERRORS	plots prediction error time series graphics.
ACF	plots prediction error autocorrelation function graphics.
PACF	plots prediction error partial autocorrelation function graphics.
IACF	plots prediction error inverse autocorrelation function graphics.
WN	plots white noise graphics.
MODELS	plots model graphics.
FORECASTS	plots forecast graphics.
MODELFORECASTSONLY	plots forecast graphics with confidence limits in the data range.
FORECASTSONLY	plots the forecast in the forecast horzion only.
LEVELS	plots smoothed level component graphics.
SEASONS	plots smoothed seasonal component graphics.
TRENDS	plots smoothed trend (slope) component graphics.
ALL	Same as specifying all of the above PLOT= options.

For example, PLOT=FORECASTS plots the forecasts for each series. The PLOT= option produces printed output for these results utilizing the Output Delivery System (ODS). The PLOT= statement is experimental for this release of SAS.

ODS Graph Names

PROC HPF 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 7.2.

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the PLOT= option in the HPF statement. For more information on the ODS GRAPHICS statement, refer to Chapter 9, "Statistical Graphics Using ODS" (*SAS/ETS User's Guide*).

Table 7.2. ODS Graphics Produced by PROC HPF

ODS Graph Name	Plot Description	Statement	PLOT= Option
DemandErrorsPlot	Average Demand Errors	PROC HPF	PLOT=ERRORS
DemandForecastsPlot	Average Demand Forecasts	PROC HPF	PLOT=FORECASTS
DemandIntervalHistogram	Demand Interval Histogram	PROC HPF	PLOT=MODELS
DemandIntervalPlot	Demand Interval Forecast Plot	PROC HPF	PLOT=MODELS
DemandSizeHistogram	Demand Size Histogram	PROC HPF	PLOT=MODELS
DemandSizePlot	Demand Size Forecast Plot	PROC HPF	PLOT=MODELS
ErrorACFNORMPlot	Standardized autocorrelation of Prediction Errors	PROC HPF	PLOT=ACF
ErrorACFPlot	Autocorrelation of Prediction Errors	PROC HPF	PLOT=ACF
ErrorHistogram	Prediction Error Histogram	PROC HPF	PLOT=ERRORS
ErrorIACFNORMPlot	Standardized inverse autocorrelation of Prediction Errors	PROC HPF	PLOT=IACF
ErrorIACFPlot	Inverse autocorrelation of Prediction Errors	PROC HPF	PLOT=IACF
ErrorPACFNORMPlot	Standardized partial autocorrelation of Prediction Errors	PROC HPF	PLOT=PACF
ErrorPACFPlot	Partial autocorrelation of Prediction Errors	PROC HPF	PLOT=PACF
ErrorPlot	Plot of Prediction Errors	PROC HPF	PLOT=ERRORS
ErrorWhiteNoiseLogProbPlot	White noise log probability plot of Prediction Errors	PROC HPF	PLOT=WN
ErrorWhiteNoiseProbPlot	White noise probability plot of Prediction Errors	PROC HPF	PLOT=WN
ForecastsOnlyPlot	Forecasts Only Plot	PROC HPF	PLOT=FORECASTONLY
ForecastsPlot	Forecasts Plot	PROC HPF	PLOT=FORECAST
LevelStatePlot	Smoothed Level State Plot	PROC HPF	PLOT=LEVELS
ModelForecastsPlot	Model and Forecasts Plot	PROC HPF	PLOT=MODELFORECAST
ModelPlot	Model Plot	PROC HPF	PLOT=MODELS

Table 7.2. (continued)

ODS Graph Name	**Plot Description**	**Statement**	**Option**
SeasonStatePlot	Smoothed Season State Plot	PROC HPF	PLOT=SEASONS
StockingAveragePlot	Stocking Average Plot	PROC HPF	PLOT=FORECASTS
StockingLevelPlot	Stocking Level Plot	PROC HPF	PLOT=FORECASTS
TrendStatePlot	Smoothed Trend State Plot	PROC HPF	PLOT=TRENDS

Examples

Example 7.1. Automatic Forecasting of Time Series Data

This example illustrates how the HPF procedure can be used for the automatic forecasting of time series data. Retail sales data is used for this illustration.

The following DATA step creates a data set from data recorded monthly at numerous points of sales. The data set, SALES, will contain a variable DATE that represents time and a variable for each sales item. Each value of the DATE variable is recorded in ascending order and the values of each of the other variables represent a single time series:

```
data sales;
   format date date9.;
   input date date9. shoes socks laces dresses coats shirts ties
         belts hats blouses;
   datalines;
   ... data lines omitted ...
   ;
run;
```

The following HPF procedure statements automatically forecast each of the monthly time series.

```
proc hpf data=sales out=nextyear;
   id date interval=month;
   forecast _ALL_;
run;
```

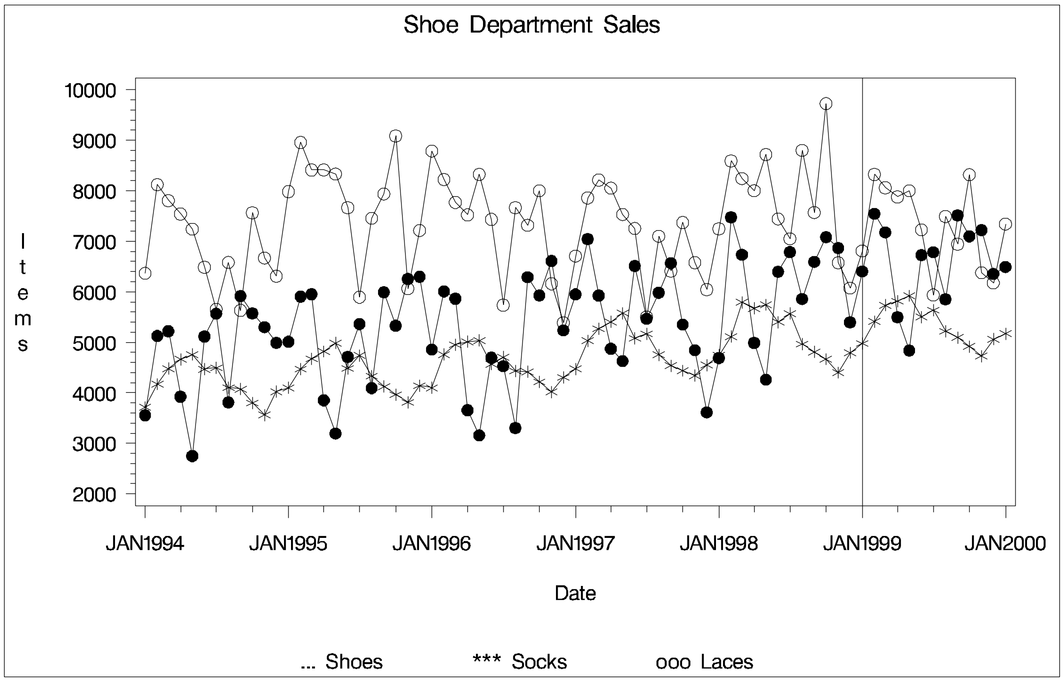
The above statements automatically select the best fitting model and generate forecasts for every numeric variable in the input data set (SALES) for the next twelve months, and stores these forecasts in the output data set (NEXTYEAR).

The following GPLOT procedure statements plot the forecasts related to shoe sales:

```
title1 "Shoe Department Sales";
axis2 label=(a=-90 r=90 "items" );
symbol1 v = dot    i = join l = 1;
symbol2 v = star   i = join l = 2;
symbol3 v = circle i = join l = 2;

proc gplot data=nextyear;
   plot shoes  * date = 1
        socks  * date = 2
        laces  * date = 3 / overlay
        haxis= '01JAN1994'd to '01DEC2000'd by year
        href=  '01JAN1999'd
        vaxis=axis2;
run;
```

The GPLOT procedure results are shown in Output 7.1.1. The historical data is shown left the horizontal reference line and the forecasts for the next twelve monthly periods is shown to the right.

Output 7.1.1. Retail Sales Forecast Plots



The following HPF procedure statements are identical to the statements above with the exception that the PRINT=FORECASTS option is specified:

```
proc hpf data=sales out=nextyear print=forecasts;
   id date interval=month;
   forecast _ALL_;
run;
```

In addition to automatically forecasting each of the monthly time series, the above statements print the forecasts using the Output Delivery System (ODS), which is partially shown in Output 7.1.2. This output shows the predictions, prediction standard errors and the upper and lower confidence limits for the next twelve monthly periods.

Output 7.1.2. Forecast Tables

```
                                  The HPF Procedure

                            Forecasts for Variable shoes

                                            Standard
 Obs        Time       Forecasts            Error       95% Confidence Limits

  62     FEB1999       7548.0041         607.5238       6357.2792       8738.7289
  63     MAR1999       7177.1472         699.4400       5806.2701       8548.0244
  64     APR1999       5497.5595         780.7609       3967.2964       7027.8227
  65     MAY1999       4838.2001         854.5169       3163.3778       6513.0224
  66     JUN1999       6728.4521         922.5244       4920.3375       8536.5668
  67     JUL1999       6786.1094         985.9738       4853.6362       8718.5826
  68     AUG1999       5853.9650        1045.6953       3804.4399       7903.4900
  69     SEP1999       7517.0144        1102.2949       5356.5561       9677.4728
  70     OCT1999       7100.2489        1156.2315       4834.0769       9366.4210
  71     NOV1999       7224.6449        1207.8618       4857.2793       9592.0106
  72     DEC1999       6357.1556        1257.4701       3892.5594       8821.7518
  73     JAN2000       6492.2657        1305.2871       3933.9500       9050.5815
```

Example 7.2. Automatic Forecasting of Transactional Data

This example illustrates how the HPF procedure can be used to automatically forecast transactional data. Internet data is used for this illustration.

The following DATA step creates a data set from data recorded at several Internet Web sites. The data set, WEBSITES, will contain a variable TIME that represents time and the variables ENGINE, BOATS, CARS, and PLANES that represent Internet Web site data. Each value of the TIME variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

```
data websites;
   format time datetime.;
   input time datetime. boats cars planes engines;
   datalines;
   ... data lines omitted ...
   ;
run;
```

The following HPF procedure statements automatically forecast each of the transactional data series:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

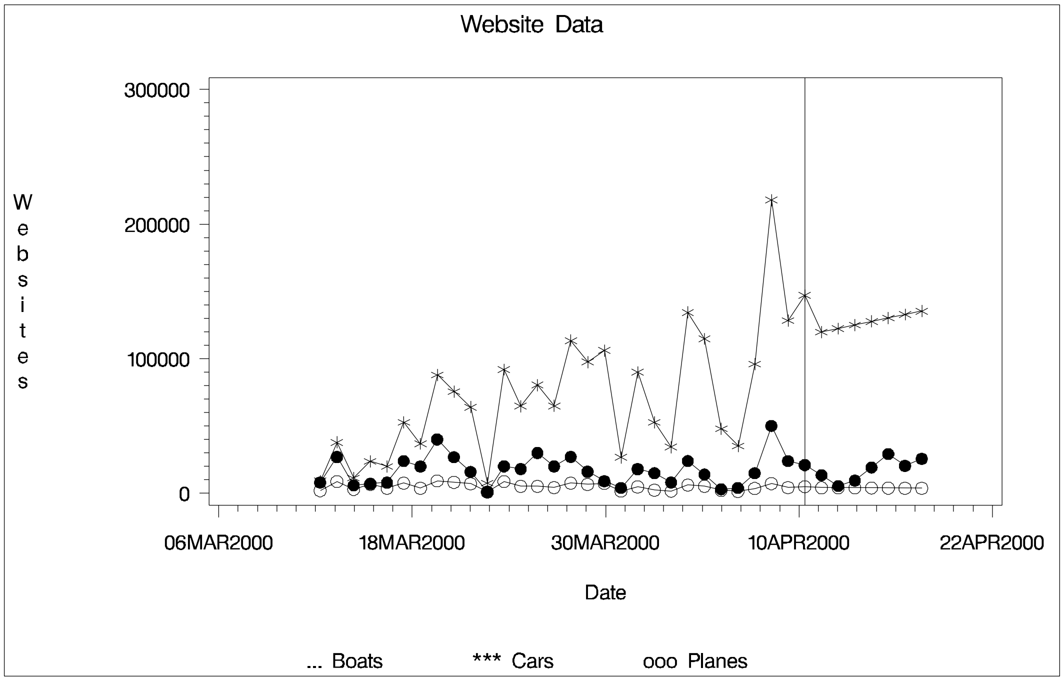
The above statements accumulate the data into a daily time series and automatically generate forecasts for the BOATS, CARS, and PLANES variables in the input data set (WEBSITES) for the next week and stores the forecasts in the output data set (NEXTWEEK).

The following GPLOT procedure statements plot the forecasts related to the Internet data:

```
title1 "Website Data";
axis2 label=(a=-90 r=90 "Websites" );
symbol1 v = dot    i = join l = 1;
symbol2 v = star   i = join l = 2;
symbol3 v = circle i = join l = 2;

proc gplot data=nextweek;
   plot boats  * time = 1
        cars   * time = 2
        planes * time = 3 / overlay
        haxis= '13MAR2000:00:00:00'dt to
               '18APR2000:00:00:00'dt by dtweek
        href=  '11APR2000:00:00:00'dt
        vaxis=axis2;
run;
```

The GPLOT procedure results are shown in Output 7.2.1. The historical data is shown to the left of the horizontal reference line and the forecasts for the next twelve monthly periods are shown to the right.

Output 7.2.1. Internet Data Forecast Plots



Example 7.3. Specifying the Forecasting Model

In the previous example, the HPF procedure was used to automatically select the appropriate forecasting model using the root mean square error (RMSE) as the default selection criterion. This example illustrates how the HPF procedure can be used to more narrowly specify the possible candidate models. Internet data from the previous example are used for this illustration.

In this example, we will forecast the BOATS variable using the best seasonal forecasting model (BESTS) that minimizes the mean absolute percent error (MAPE), the CARS variable using the best nonseasonal forecasting model (BESTN) that minimizes the mean square error (MSE) using holdout sample analysis, and the PLANES variable using Log Winters (additive). The following HPF procedure statements forecast each of the transactional data series based on these requirements:

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats   / model=bests select=mape;
   forecast cars    / model=bestn select=mse holdout=5;
   forecast planes  / model=addwinters transform=log;
run;
```

Example 7.4. Extending the Independent Variables for Multivariate Forecasts

In the previous example, the HPF procedure was used to forecast several transactional series variables using univariate models. This example illustrates how the HPF procedure can be used to extend the independent variables associated with a multiple regression forecasting problem. Specifically, PROC HPF is used to extend the independent variables for use in forecasting a regression model.

In this example, we will accumulate and forecast the BOATS, CARS, and PLANES variables as illustrated in the previous example. In addition, we will accumulate the ENGINES variable to form a time series that is then extended with missing values within the forecast horizon with the specification of MODEL=NONE.

```
proc hpf data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast engines / model=none;
   forecast boats   / model=bests select=mape;
   forecast cars    / model=bestn select=mse holdout=5;
   forecast planes  / model=winters transform=log;
run;
```

The following AUTOREG procedure statements are used to forecast the ENGINES variable by regressing on the independent variables (BOATS, CARS, and PLANES).

```
proc autoreg data= nextweek;
   model engines = boats cars planes / noprint;
   output out=enginehits p=predicted;
run;
```

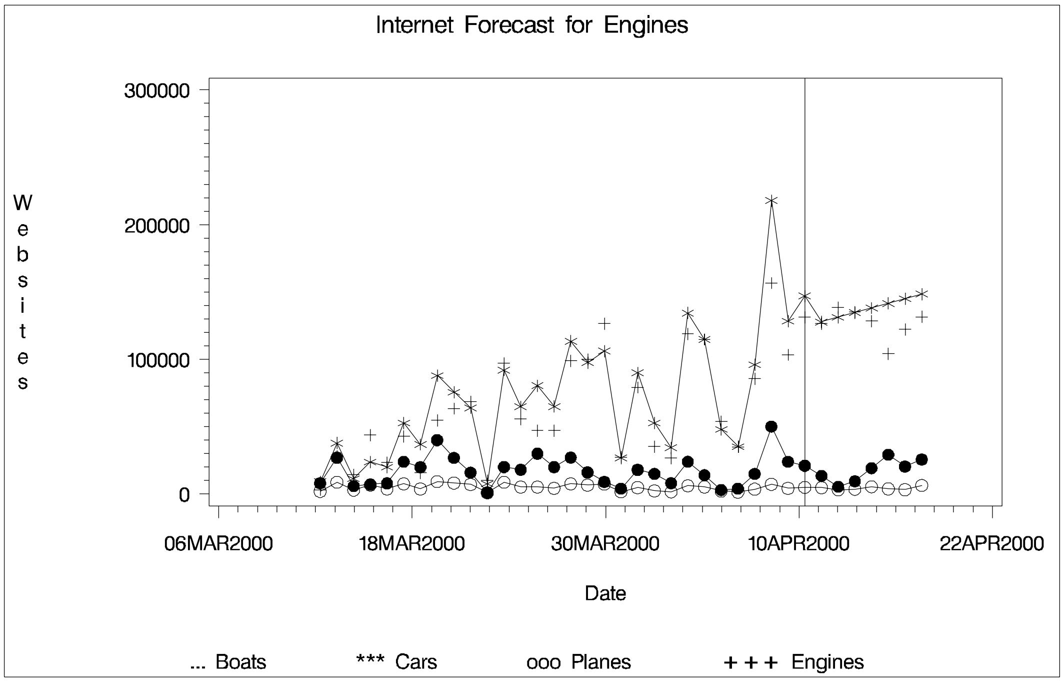
The output data set (NEXTWEEK) of the PROC HPF statement is used as an input data set for the PROC AUTOREG statement. The output data set of PROC AUTOREG contains the forecast of the variable ENGINES based on the regression model with the variables BOATS, CARS, and PLANES as regressors. See the AUTOREG procedure for details on autoregression models.

The following GPLOT procedure statements plot the forecasts related to the ENGINES variable:

```
proc gplot data=enginehits;
   plot boats  * time = 1
        cars   * time = 2
        planes * time = 3
        predicted * time = 4 / overlay
        haxis= '13MAR2000:00:00:00'dt to
               '18APR2000:00:00:00'dt by dtweek
        href=  '11APR2000:00:00:00'dt
        vaxis=axis2;
run;
```

The GPLOT procedure results are shown in Output 7.4.1. The historical data is shown left the horizontal reference line and the forecasts for the next four weekly periods is shown to the right.

Output 7.4.1. Internet Data Forecast Plots



Example 7.5. Forecasting Intermittent Time Series Data

This example illustrates how the HPF procedure can be used to forecast intermittent time series data. Inventory demand is used for this illustration.

The following DATA step creates a data set from inventory data recorded at no particular frequency. The data set, INVENTORY, will contain a variable DATE that represents time and the demand variables (TIRES, HUBCAPS, and LUGBOLTS), which represent inventory items. Each value of the DATE variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

```
data inventory;
   format date date9.;
   input date date9. tires hubcaps lugbolts;
   datalines;
   ... data lines omitted ...
   ;
run;
```

The following HPF procedure statements forecast each of the transactional data series using and intermittent demand model:

```
proc hpf data=inventory out=nextmonth lead=4 print=forecasts;
   id date interval=week accumulate=total;
   forecast tires hubcaps lugbolts / model=idm;
run;
```

The above statements accumulate the data into a weekly time series, and generate forecasts for the TIRES, HUBCAPS, and LUGBOLTS variables in the input data set (INVENTORY) for the four weekly periods, and store the forecasts in the output data set (NEXTMONTH). The PRINT=FORECAST option produces the results partially shown in Output 7.5.1. The first table records the demand series and predictions. The second table represents forecasts or recommended stocking levels.

Output 7.5.1. Forecast Tables

```
                              The HPF Procedure

                          Demands for Variable tires

                          Demand      Demand          Estimate of Mean
         Demand          Intervals     Size           Demand per Period
Index              Time     Actual     Actual     Actual     Predict         Std

    1   Sun, 31 Aug 1997        14     6.0000    0.42857    0.42857      .
    2   Sun, 26 Oct 1997         8     4.0000    0.50000    0.42857    0.15082
    3    Sun, 1 Mar 1998        18     2.0000    0.11111    0.43002    0.15082
    4   Sun, 26 Apr 1998         8     2.0000    0.25000    0.42103    0.15082
    5   Sun, 31 May 1998         5     2.0000    0.40000    0.41593    0.15082
    6   Sun, 27 Sep 1998        17     6.0000    0.35294    0.41497    0.15082
    7    Sun, 3 Jan 1999        14      .          .        0.41276    0.15082

Stock = (Interval Actual)*(Predict) - (Size Actual)

                          Demands for Variable tires

         Demand              Estimate of Mean Demand per Period
   Index              Time   95% Confidence Limits        Error         Stock

      1   Sun, 31 Aug 1997     .            .          0.0000000     -0.000000
      2   Sun, 26 Oct 1997   0.13296      0.72418      0.0714286     -0.571429
      3    Sun, 1 Mar 1998   0.13441      0.72563      -.3189115      5.740406
      4   Sun, 26 Apr 1998   0.12542      0.71663      -.1710252      1.368201
      5   Sun, 31 May 1998   0.12033      0.71154      -.0159339      0.079670
      6   Sun, 27 Sep 1998   0.11936      0.71058      -.0620274      1.054465
      7    Sun, 3 Jan 1999   0.11716      0.70837       .              .

   Stock = (Interval Actual)*(Predict) - (Size Actual)


                         Forecasts for Variable tires

                                             Standard
  Obs               Time      Forecasts        Error      95% Confidence Limits

   84     Sun, 3 Jan 1999       0.41276      0.15082       0.11716      0.70837
   85    Sun, 10 Jan 1999       0.41276      0.15082       0.11716      0.70837
   86    Sun, 17 Jan 1999       0.41276      0.15082       0.11716      0.70837
   87    Sun, 24 Jan 1999       0.41276      0.15082       0.11716      0.70837
```

Example 7.6. Illustration of ODS Graphics (Experimental)

This example illustrates the use of experimental ODS graphics.

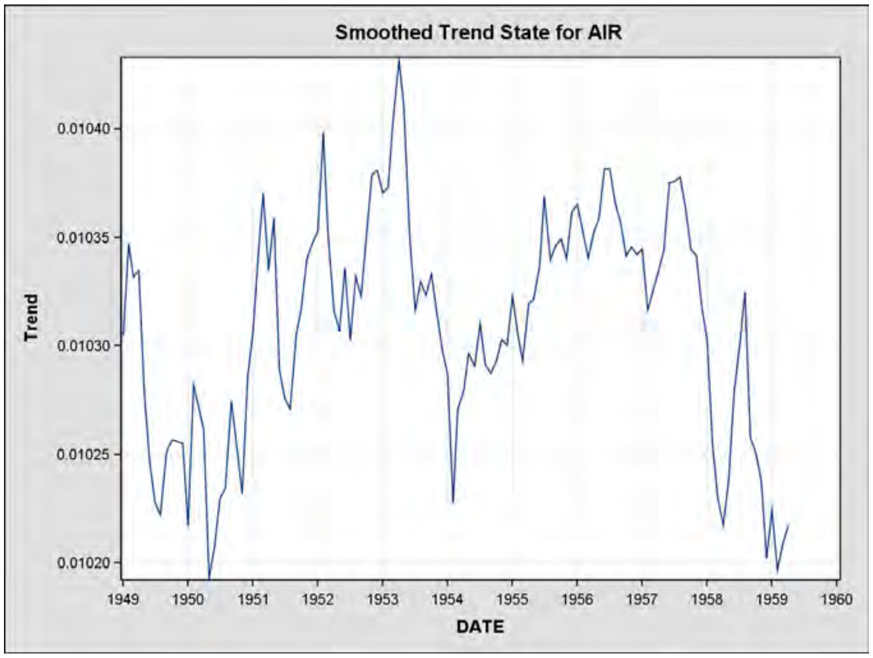
The following statements utilize the SASHELP.AIR data set to automatically forecast the time series of international airline travel.

The graphical displays are requested by specifying the experimental ODS GRAPHICS statement and the experimental PLOT= option in the PROC HPF statement. In this case, all plots are requested. Output 7.6.1 through Output 7.6.4 show a selection of the plots created.

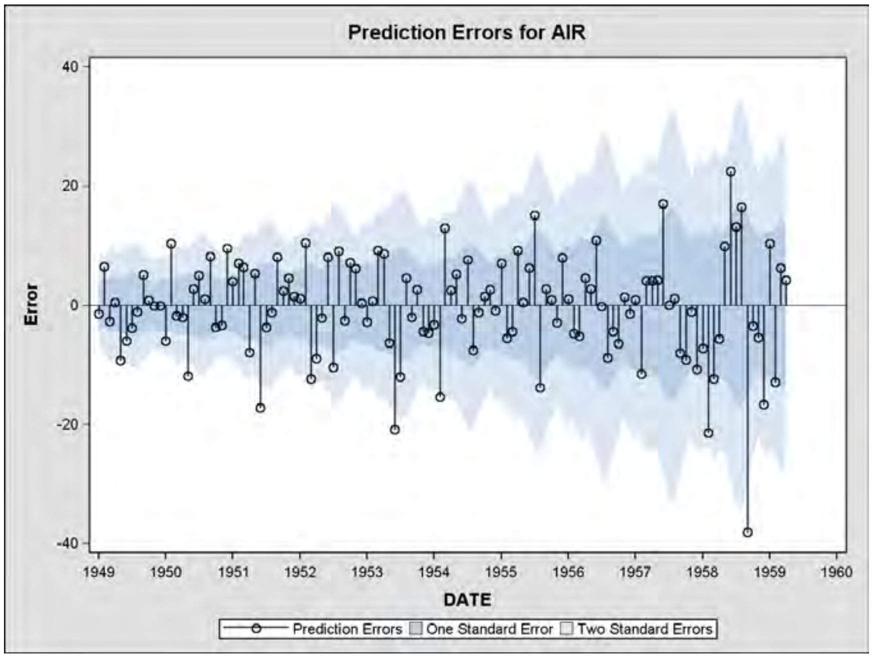
For general information about ODS graphics, refer to Chapter 9, "Statistical Graphics Using ODS" (*SAS/ETS User's Guide*). For specific information about the graphics available in the HPF procedure, see the "ODS Graphics" section on page 257.

```
ods html;
ods graphics on;

proc hpf data=sashelp.air out=_null_ lead=20 back=20 print=all plot=all;
   id date interval=month;
  forecast air / model=best transform=auto select=mape;
run;

ods graphics off;
ods html close;
```

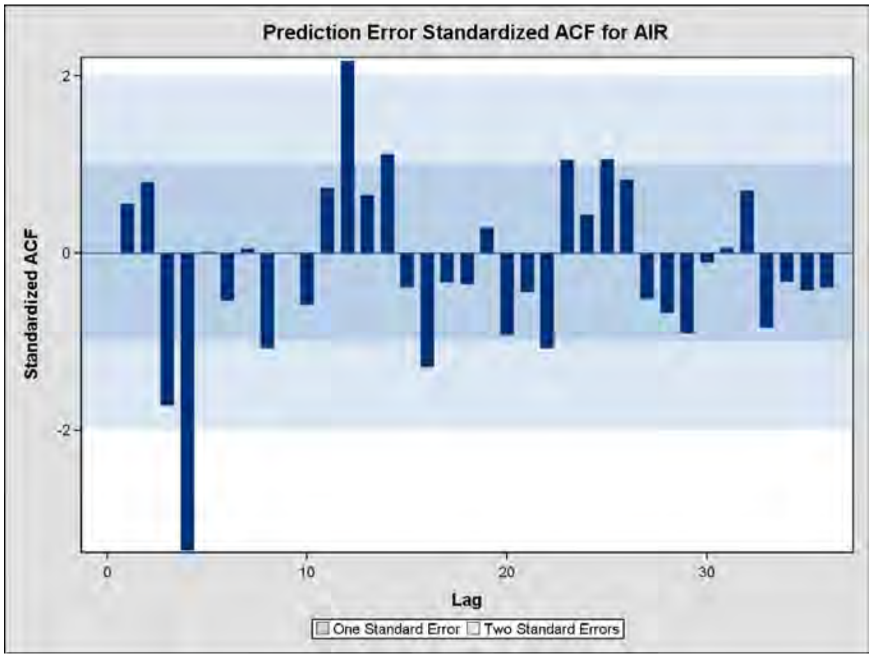
Output 7.6.1. Smoothed Trend Plot (Experimental)



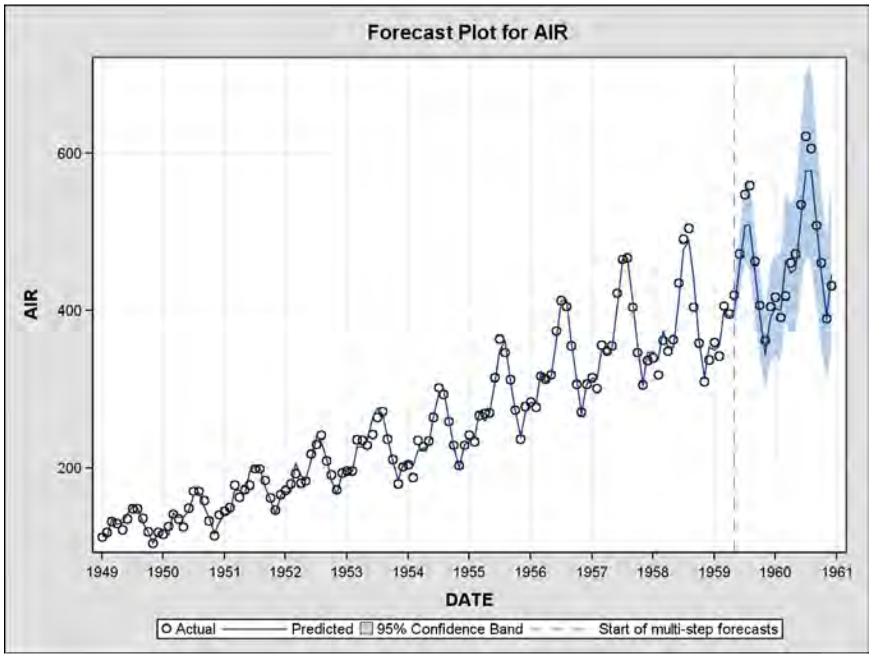
Output 7.6.2. Prediction Error Plot (Experimental)



Output 7.6.3. Prediction Error Standardized ACF Plot (Experimental)



Output 7.6.4. Forecast Plot (Experimental)



References

Pyle, D. (1999), *Data Preparation for Data Mining*, San Francisco: Morgan Kaufman Publishers, Inc.

Chapter 8
The HPFARIMASPEC Procedure

Chapter Contents

Chapter 8
The HPFARIMASPEC Procedure

Overview

The HPFARIMASPEC procedure is used to create an ARIMA model specification file. The output of this procedure is an XML file that stores the intended ARIMA model specification. This XML specification file can be used for different purposes; for example, to populate the model repository used by the HPFENGINE procedure (see Chapter 10, “The HPFENGINE Procedure”). You can specify very general ARIMA models using this procedure. In particular, any model that can be analyzed using the ARIMA procedure can be specified; see Chapter 11, “The ARIMA Procedure” (*SAS/ETS User’s Guide*). Moreover, the model specification can include series transformations such as log or Box-Cox transformations.

Getting Started

The following example shows how to create an ARIMA model specification file. In this example the specification for an Airline model with one input is created.

```
proc hpfarimaspec repository=work.arima
                  name=Airline1
                  label="Airline model with one input";
   forecast symbol=Y q=(1)(12) dif=(1, 12) noint
            transform=log;
   input symbol=X dif=(1, 12);
   estimate method=ml;
run;
```

The options in the PROC HPFARIMASPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in the catalog SASUSER.ARIMA, the NAME= option specifies that the name of the file be Airline1.xml, and the LABEL= option specifies a label for this catalog member. The other statements in the procedure specify the ARIMA model and the options used to control the parameter estimation process for the model. The model specification begins with the FORECAST statement that specifies the following:

- transformation, such as log or Box-Cox, and the differencing orders associated with the variable that is to be forecast
- autoregressive and moving-average polynomials
- presence or absence of the constant in the model

Here, according to the FORECAST statement, the model contains no constant term and has a two-factor moving-average polynomial of orders 1 and 12. The forecast variable is log transformed and differenced with differencing orders 1 and 12. The SYMBOL= option in the FORECAST statement can be used to provide a convenient name for the forecast variable. This name is only a placeholder, and a proper data variable will be associated with this name when this model specification is used in actual data analysis.

Next, the INPUT statement provides the transfer function specification associated with the input variable in the model. In the INPUT statement you can specify

- transformation, such as log or Box-Cox, and the lagging and differencing orders associated with the input variable
- numerator and denominator polynomials associated with the transfer function input

In this case the input variable is differenced with differencing orders 1 and 12, and it enters the model as a simple regressor. Here again the SYMBOL= option can be used to supply a convenient name for the input variable. If a model contains multiple input variables then each input variable specification has to be given using a separate INPUT statement.

Lastly, the ESTIMATE statement specifies that the model be estimated using the ML method of estimation.

Syntax

The HPFARIMASPEC procedure uses the following statements.

PROC HPFARIMASPEC *options*;
  FORECAST *options* ;
  INPUT *options*;
  ESTIMATE *options*;

Functional Summary

The statements and options controlling the HPFARIMASPEC procedure are summarized in the following table.

Description	Statement	Option
Model Repository Options		
specify the model repository	PROC HPFARIMASPEC	REPOSITORY=
specify the model specification name	PROC HPFARIMASPEC	NAME=

Description	**Statement**	**Option**
specify the model specification label	PROC HPFARIMASPEC	LABEL=
Options for Specifying Symbolic Series Names		
specify a symbolic name for the response series	FORECAST	SYMBOL=
specify a symbolic name for the input series	INPUT	SYMBOL=
specify a predefined trend as the input series	INPUT	PREDEFINED=
Options for Specifying the Model		
specify the response series transformation	FORECAST	TRANSFORM=
specify the response series differencing orders	FORECAST	DIF=
specify the input series transformation	INPUT	TRANSFORM=
specify the input series differencing orders	INPUT	DIF=
specify the input series lagging order	INPUT	DELAY=
specify the ARIMA part of the model	FORECAST	
specify the AR polynomial	FORECAST	P=
specify autoregressive starting values	FORECAST	AR=
specify the MA polynomial	FORECAST	Q=
specify moving average starting values	FORECAST	MA=
indicate absence of a constant in the model	FORECAST	NOINT
specify a starting value for the mean parameter	FORECAST	MU=
specify the NOISE variance	FORECAST	NOISEVAR=
specify the transfer function part of the model	INPUT	
specify the numerator polynomial of a transfer function	INPUT	NUM=
specify starting values for the numerator polynomial coefficients	INPUT	NC=
specify starting value for the zero degree numerator polynomial coefficient	INPUT	NZ=
specify the denominator polynomial of a transfer function	INPUT	DEN=
specify starting values for the denominator polynomial coefficients	INPUT	DC=
Options to Control the Parameter Estimation		
specify the estimation method	ESTIMATE	METHOD=
suppress the iterative estimation process	ESTIMATE	NOEST
specify the maximum number of iterations	ESTIMATE	MAXITER=
specify convergence criterion	ESTIMATE	CONVERGE=

PROC HPFARIMASPEC Statement

PROC HPFARIMASPEC *options;*

The following options can be used in the PROC HPFARIMASPEC statement:

LABEL= *SAS-label*
specfies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

NAME= *SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

REPOSITORY= *SAS-catalog-name*
REPOSITORY= *SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

FORECAST Statement

FORECAST *options;*

The FORECAST statement specifies the operations to be performed on the response series as well as the auto regressive and moving average polynomials in the model. The presence or absence of a constant in the model is also specified here.

The following options are used in the FORECAST statement.

(SYMBOL|VAR)= *variable*
specifies a symbolic name for the dependent series. This symbol specification is optional. If the SYMBOL= option is not specified, *Y* is used as a default symbol.

DIF= *order*
DIF= *(order1, order2, ...)*
specifies the differencing orders for the dependent series. For example, DIF= (1 12) specifies that the series be differenced using the operator $(1 - B)(1 - B^{12})$. The differencing orders can be positive integers or they can be “s”, indicating a placeholder that will be substituted later with an appropriate value. The use of placeholders is explained further in Example 8.3.

P= *order*
P= (*lag, ..., lag***) ... (***lag, ..., lag***)**
P= (*lag, ..., lag***)<**s_1**> ... (***lag, ..., lag***)<**s_k**>**
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

$$(1 - \phi_{1,1}B - \phi_{1,2}B^2 - \phi_{1,3}B^5)(1 - \phi_{2,1}B^6 - \phi_{2,2}B^{12})$$

Optionally, you can specify *multipliers* after the parenthesized lists. For example, P=(1)(1)12 is equivalent to P=(1)(12), and P=(1,2)4(1)12(1,2)24 is equivalent to P=(4,8)(12)(24,48). These multipliers can either be positive integers or they can be "s", indicating a placeholder that will be substituted later with an appropriate value. The use of placeholders in the multiplier specification is explained in Example 8.3.

Q= *order*
Q= (*lag, ..., lag***) ... (***lag, ..., lag***)**
Q= (*lag, ..., lag***)<**s_1**> ... (***lag, ..., lag***)<**s_k**>**

specifies the moving-average part of the model. By default, no moving average parameters are fit.

The manner of specification of the moving-average part is identical to the specification of the autoregressive part described in the P= option.

AR= *value* **...**

lists starting values for the autoregressive parameters.

MA= *value* **...**

lists starting values for the moving-average parameters.

NOCONSTANT
NOINT

suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter μ is omitted.)

MU= *value*

specifies the MU parameter.

NOISEVAR= *value*

specifies the noise variance. This is only useful if you want to specify an externally published model that is fully specified.

TRANSFORM= *option*

specifies the transformation to be applied to the time series. The following transformations are provided:

NONE	No transformation applied
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 intended time series must be strictly positive.

INPUT Statement

INPUT *options;*

The INPUT statements specify the transfer function inputs in the model. A separate INPUT statement is needed for each of the transfer function inputs. In this statement you can specify all the features of the transfer function associated with the input variable under consideration. The following options are used in the INPUT statement.

(SYMBOL|VAR)= *variable*

specifies a symbolic name for the dependent series. This symbol specification is optional. If the SYMBOL= option or the PREDEFINED= option is not specified then *X* is used as a default symbol. If there are multiple INPUT statements then an attempt is made to generate a unique set of input symbols.

PREDEFINED= *option*

associates a predefined trend or a set of seasonal dummy variables with this transfer function. The SYMBOL= and PREDEFINED= options are mutually exclusive.

In the following list of options, let t represent the observation count from the start of the period of fit for the model, and let X_t be the value of the time trend variable at observation t.

LINEAR	A linear trend, with $X_t = t - c$
QUADRATIC	A quadratic trend, with $X_t = (t - c)^2$
CUBIC	A cubic trend, with $X_t = (t - c)^3$
INVERSE	An inverse trend, with $X_t = 1/t$
SEASONAL	Seasonal dummies. 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 do not include an intercept term. Each element of a seasonal dummy regressor is either zero or one, based on the following rule: $X_{i,t} = \begin{cases} 1 & \text{when } i = t \\ 0 & \text{otherwise} \end{cases}$ 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.

DIF= *order*
DIF= *(order1, order2, ...)*

specifies the differencing orders for the input series. See the DIF= option of the FORECAST statement for additional information.

DELAY= *order*

specifies the delay, or lag, order for the input series.

NUM= *order*
NUM= (*lag, ..., lag***) ... (***lag, ..., lag***)**
NUM= (*lag, ..., lag***)<**s_1**> ... (***lag, ..., lag***)<**s_k**>**

specifies the numerator polynomial of the transfer function. See the P= option of the FORECAST statement for additional information concerning the polynomial order specification.

DEN= *order*
DEN= (*lag, ..., lag***) ... (***lag, ..., lag***)**
DEN= (*lag, ..., lag***)<**s_1**> ... (***lag, ..., lag***)<**s_k**>**

specifies the denominator polynomial of the transfer function. See the P= option of the FORECAST statement for additional information concerning the polynomial order specification.

NC= *value* **...**

lists starting values for the numerator polynomial coefficients.

DC= *value* **...**

lists starting values for the denominator polynomial coefficients.

NZ= *value*

specifies the scale parameter, i.e., the zero degree coefficient of the numerator.

TRANSFORM= *option*

specifies the transformation to be applied to the time series. The following transformations are provided:

NONE	No transformation applied
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 intended time series must be strictly positive.

ESTIMATE Statement

ESTIMATE *options;*

This is an optional statement in the procedure. Here you can specify the estimation method or whether to hold the model parameters fixed to their starting values. You can also specify some parameters that control the nonlinear optimization process. The following options are available.

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.

NOEST

uses the values specified with the AR=, MA=, . . . , etc. as final parameter values. The estimation process is suppressed except for the estimation of the residual variance. The specified parameter values are used directly by the next FORECAST statement. Use of NOEST requires that all parameters be specified via the AR=, MA=, . . . , etc. Partially specified models will cause an error when used by the HPFENGINE procedure. 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.)

CONVERGE= *value*

specifies the convergence criterion. Convergence is assumed when the largest change in the estimate for any parameter is less than 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.

MAXITER= n
MAXIT= n

specifies the maximum number of iterations allowed. The default is MAXITER=50.

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.

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.

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.

Examples

Example 8.1. Some Syntax Illustrations

The following code fragments illustrate the HPFARIMASPEC syntax for some of the commonly needed modeling activities. Suppose that a variety of ARIMA models are to be fit to a data set that contains a sales series as the forecast variable and several promotional events as predictor series. In all these cases the model repository is kept the same, sasuser.arima, and the models are named as *model1, model2, ...*, to ensure uniqueness. Note that in a given repository, the models must have unique names. The symbols for the forecast and input variables are *sales* and *promo1, promo2, ...*, respectively.

```
 /* Two transfer functions */
 proc hpfarimaspec repository=work.arima
                   name=model1;
    forecast symbol=sales transform=log
             q=(1)(12) dif=(1,12) noint;
    input symbol=promo1 dif=(1, 12) den=2;
    input symbol=promo2 num=2 delay=3;
 run;


/* Box-Cox transform and Estimation Method=ML */
proc hpfarimaspec
        repository=work.arima
        name=model2;
    forecast symbol=sales transform=BoxCox(0.8)
        p=2;
    estimate method=ml;
run;


/* suppress parameter estimation: in this    */
/* case all the parameters must be specified */
proc hpfarimaspec repository=work.arima
                  name=model3;
   forecast symbol=sales transform=log
            p=2 ar=0.1 0.8 mu=3.5;
   estimate noest method=ml;
run;


/* Supply starting values for the parameters */
proc hpfarimaspec repository=work.arima
                  name=model4;
   forecast symbol=sales transform=log
            p=2 ar=0.1 0.8 mu=3.5;
   input symbol=promo1
         den=1 dc=0.1 nz=-1.5;
run;
```

```
/* Create a generic seasonal Airline model with one input
   that is applicable for different season lengths
*/
proc hpfarimaspec
        repository=work.arima
        name=model5
        label="Generic Airline Model with One Input";
   forecast symbol=Y q=(1)(1)s dif=(1, s) noint
        transform= log;
   input symbol=X dif=(1, s);
run;
```

Example 8.2. How to Include ARIMA Models in a Model Selection List

One of the primary uses of the HPFARIMASPEC procedure is to add candidate ARIMA models to a model selection list that can be used by the HPFENGINE procedure (see Chapter 10, "The HPFENGINE Procedure"). The HPFARIMASPEC procedure is used to create the ARIMA model specifications and the HPFSELECT procedure is used to add the specifications to a model selection list (see Chapter 16, "The HPFSELECT Procedure"). This example illustrates this scenario.

Here the Gas Furnace Data, "Series J" from Box and Jenkins (1976), is used. This data set contains two series, Input Gas Rate and Output CO2. The goal is to forecast the output CO2, using the input Gas Rate as a predictor if necessary.

The following DATA step statements read the data in a SAS data set.

```
data seriesj;
   input GasRate CO2 @@;
   datalines;
    -0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5
    0.373  53.4  0.441  53.1  0.461  52.7  0.348  52.4
    /* -- data lines --                         --*/
    0.034  57.0  0.204  58.0  0.253  58.6  0.195  58.5
    0.131  58.3  0.017  57.8 -0.182  57.3 -0.262  57.0
   ;
```

Three candidate models are specified, *m1, m2*, and *m3*. Out of these three models, *m1* is known to be a good fit to the data. It is a transfer function model involving the input Gas Rate. The other two models are simplified versions of *m1*. The following syntax shows how to specify these models and how to create a selection list that combines them using the HPFSELECT procedure. In the HPFSELECT procedure note the use of the INPUTMAP option in the SPEC statement. It ties the symbolic variable names used in the HPFARIMASPEC procedure with the actual variable names in the data set. If the symbolic names were appropriate to start with, then the INPUTMAP option need not be used.

```
*make spec1;
proc hpfarimaspec repository=work.mycat
                  name=m1;
   forecast symbol=y p=2;
   input symbol=x delay=3 num=(1,2) den=1;
   estimate method=ml;
run;

*make spec2;
proc hpfarimaspec repository=work.mycat
                  name=m2;
   forecast symbol=y p=2;
   input symbol=x delay=3;
   estimate method=ml;
run;

*make spec3;
proc hpfarimaspec repository=work.mycat
                  name=m3;
   forecast symbol=y p=2;
   estimate method=ml;
run;

*make a selection list that includes m1, m2 and m3;
proc hpfselect repository=work.mycat
               name=myselect;

   spec m1 / inputmap(symbol=y var=co2)
             inputmap(symbol=x var=gasrate);

   spec m2 / inputmap(symbol=y var=co2)
             inputmap(symbol=x var=gasrate);

   spec m3 / inputmap(symbol=y var=co2);
run;
```

This selection list can now be used in the HPFENGINE procedure for various types of analyses. The following syntax shows how to compare these models based on the default comparison criterion, Mean Absolute Percentage Error (MAPE). As expected, model *m1* turns out to be the best of the three compared (see Output 8.2.1).

```
proc hpfengine data=seriesj
               repository=work.mycat
               globalselection=myselect
               lead=0
               print=(select);
   forecast co2;
   input    gasrate;
run;
```

Output 8.2.1. Model Selection Based on the MAPE Criterion

```
                    Model          MAPE    Selected

                      M1      0.31478457      Yes
                      M2      0.50671996      No
                      M3      0.53295590      No
```

Example 8.3. How to Create a Generic Seasonal Model Spec That Is Suitable for Different Season Lengths

In the case of many seasonal model specifications, it is possible to describe a generic specification that is applicable in a variety of situations just by changing the season length specifications at appropriate places. As an example consider the Airline model, which is very useful for modeling seasonal data. The Airline model for a monthly series can be specified using the following syntax:

```
proc hpfarimaspec repository=work.specs
                  name=MonthlyAirline
                  label=
 "Airline Model For A Series With Season Length 12";
   forecast symbol=Y q=(1)(1)12 dif=(1, 12) noint
            transform= log;
run;
```

It is easy to see that the same syntax is applicable to a quarterly series if the multiplier in the MA specification is changed from 12 to 4 and the seasonal differencing order is similarly changed from 12 to 4. A generic specification that allows for late binding of season lengths can be generated by the following syntax:

```
proc hpfarimaspec repository=work.specs
                  name=GenericAirline
                  label="Generic Airline Model";
   forecast symbol=Y q=(1)(1)s dif=(1, s) noint
            transform= log;
run;
```

In this syntax the multiplier in the MA specification is changed from 12 to “s”, and similarly the seasonal differencing order 12 is changed to “s”. This syntax creates a template for the Airline model that is applicable to different season lengths. When the HPFENGINE procedure, which actually uses such model specifications to estimate the model and produce the forecasts, encounters such “generic” specification it automatically creates a proper specification by replacing the placeholders for the seasonal multiplier and the seasonal differencing order with the value implied by the ID variable or its SEASONALITY= option. The following example illustrates the use of this generic spec. It shows how the same spec can be used for monthly and quarterly series. The parameter estimates for monthly and quarterly series are given in Output 8.3.1 and Output 8.3.2, respectively.

```
/* Create a selection list that contains
   the Generic Airline Model */
proc hpfselect repository=work.specs
               name=genselect;
   spec GenericAirline;
run;


/* Monthly interval */
proc hpfengine data=sashelp.air
               repository=work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=month;
   forecast air;
run;
```

Output 8.3.1. Parameter Estimates for the Monthly Series

```
                           The HPFENGINE Procedure

                             Parameter Estimates

                                               Standard                    Approx
Component    Parameter        Estimate            Error     t Value      Pr > |t|

AIR          MA1_1             0.37727          0.08196        4.60        <.0001
AIR          MA2_12            0.57236          0.07802        7.34        <.0001
```

```
/* Create a quarterly series to illustrate
   accumulating the monthly Airline series to quarterly*/
proc timeseries data=sashelp.air out=Qair;
   id date interval=quarter;
   var air / accumulate=total;
run;


/* Quarterly interval */
proc hpfengine data=Qair
               repository= work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=quarter;
   forecast air;
run;
```

Output 8.3.2. Parameter Estimates for the Quarterly Series

```
                             The HPFENGINE Procedure

                               Parameter Estimates

                                                   Standard                  Approx
  Component     Parameter       Estimate             Error     t Value     Pr > |t|

  AIR           MA1_1            0.05892           0.15594        0.38       0.7075
  AIR           MA2_4            0.50558           0.14004        3.61       0.0008
```

References

Box, G. E. P. and Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control,* San Francisco: Holden-Day.

Chapter 9
The HPFDIAGNOSE Procedure

Chapter Contents

Chapter 9
The HPFDIAGNOSE Procedure

Overview

The HPFDIAGNOSE procedure provides a comprehensive set of tools for automated univariate time series model identification. Time series data can have outliers, structural changes, and calendar effects. In the past, finding a good model for time series data usually required experience and expertise in time series analysis.

The HPFDIAGNOSE procedure automatically diagnoses the statistical characteristics of time series and identifies appropriate models. The models that HPFDIAGNOSE considers for each time series include ARIMAX, Exponential Smoothing, and Unobserved Components models. Log transformation and stationarity tests are automatically performed. The ARIMAX model diagnostics find the AR and MA orders, detect outliers, and select the best input variables. The Unobserved Components Model diagnostics find the best components and select the best input variables.

The HPFDIAGNOSE procedure provides the following functionality:

- intermittency (or interrupted series) test
- functional transformation test
- simple differencing and seasonal differencing tests
- tentative simple ARMA order identification
- tentative seasonal ARMA order identification
- outlier detection
- significance test of events (indicator variables)
- transfer function identification
  - intermittency test
  - functional transformation for each regressor
  - simple differencing order and seasonal differencing order for each regressor
  - time delay for each regressor
  - simple numerator and denominator polynomial orders for each regressor
- intermittent demand model (automatic selection)
- exponential smoothing model (automatic selection)
- unobserved components model (automatic selection)

Getting Started

This section outlines the use of the HPFDIAGNOSE procedure and shows examples of how to create ARIMA, ESM, and UCM model specifications.

The following example prints the diagnostic tests of an ARIMA model. In the HPFDIAGNOSE statement, the SEASONALITY=12 option specifies the length of the seasonal cycle of the time series, and the PRINT=SHORT option prints the chosen model specification. The FORECAST statement specifies the dependent variable (AIR). The ARIMAX statement specifies that an ARIMA model is to be diagnosed.

```
proc hpfdiag data=sashelp.air seasonality=12 print=short;
   forecast air;
   arimax;
run;
```

Figure 9.1 shows the ARIMAX model specification. The log transformation test and trend test are conducted by default. The log transformation was applied to the dependent series and the seasonal ARIMA $(1,1,0)(0,1,1)_{12}$ model was selected. The default model selection criterion (RMSE) was used. The STATUS column explains warnings or errors during diagnostic tests. STATUS=OK indicates that the model was successfully diagnosed.

```
                            The HPFDIAGNOSE Procedure

                            ARIMA Model Specification

         Functional                                                Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

AIR      LOG        NO        1  1  0  0  1  1          12 RMSE        10.8353

                            ARIMA Model Specification

                            Variable Status

                            AIR      OK
```

Figure 9.1. ARIMAX Specification

The following example prints the diagnostic tests of an ESM for airline data. The ID statement INTERVAL=MONTH option specifies an implied seasonality of 12. The ESM statement specifies that an ESM model is to be diagnosed.

```
proc hpfdiag data=sashelp.air print=short;
   id date interval=month;
   forecast air;
   esm;
run;
```

Figure 9.2 shows the ESM model specification. The chosen model specification applied the log transformation and selected a multiplicative seasonal model with a trend component (WINTERS).

```
                              The HPFDIAGNOSE Procedure

                      Exponential Smoothing Model Specification

            Functional    Selected                             Model
Variable    Transform     Model         Component    Criterion     Statistic

AIR         LOG           WINTERS       LEVEL        RMSE            10.6521
                                        TREND
                                        SEASONAL
```

Figure 9.2. ESM Specification

The following example prints the diagnostic tests of an UCM for airline data. The UCM statement specifies that a UCM model is to be diagnosed.

```
proc hpfdiag data=sashelp.air print=short;
   id date interval=month;
   forecast air;
   ucm;
run;
```

When the column SELECTED=YES, the component is significant. When the column SELECTED=NO, the component is insignificant in Figure 9.3.

When SELECTED=YES, the STOCHASTIC column has either YES or NO. STOCHASTIC=YES indicates a component has a statistically significant variance, indicating the component is changing over time; STOCHASTIC=NO indicates the variance of a component is not statistically significant, but the component itself is still significant.

Figure 9.3 shows that the irregular, level, slope, and seasonal components are selected. The irregular, level, and seasonal components have statistically significant variances. The slope component is constant over the time.

```
                              The HPFDIAGNOSE Procedure

                    Unobserved Components Model(UCM) Specification

          Functional                                                    Model
Variable  Transform    Component    Selected  Stochastic  Seasonality  Criterion

AIR       LOG          IRREGULAR    YES       YES                      RMSE
                       LEVEL        YES       YES
                       SLOPE        YES       NO
                       SEASON       YES       YES                  12

                           Unobserved Components Model(UCM)
                                    Specification

                    Variable  Statistic    Status

                    AIR         10.9801    OK
```

Figure 9.3. Select Components

The following example shows how to pass a model specification created by the HPFDIAGNOSE procedure to the HPFENGINE procedure.

An ARIMAX model specification file, a model selection list, and a model repository SASUSER.MYCAT are created by the HPFDIAGNOSE procedure. The ARIMAX model specification file and the model selection list are contained in the SASUSER.MYCAT repository.

The OUTEST= data set is used to transmit the diagnostic results to the HPFENGINE procedure by the INEST= option. The WORK.EST_ONE data set contains the information about the data set variable and the model selection list.

```
proc hpfdiag data=sashelp.air outest=est_one
     modelrepository=sasuser.mycat criterion=MAPE;
   id date interval=month;
   forecast air;
   arimax;
run;



proc hpfengine data=sashelp.air print=(select)
     modelrepository=sasuser.mycat inest=est_one;
   forecast air;
   id date interval=month;
run;
```

Figure 9.4 shows the DIAG0 model specification created by the HPFDIAGNOSE procedure in the previous example. The model specification is labeled DIAG0 because the HPFDIAGNOSE procedure uses BASENAME=DIAG by default. The model selection list is labeled DIAG1 which can be seen in the WORK.EST_ONE data set.

```
                         The HPFENGINE Procedure

                            Model Selection
                           Criterion = MAPE

                     Model     Statistic    Selected

                     diag0     2.9422734    Yes

                     Model Selection Criterion = MAPE

     Model    Label

     diag0    ARIMA: Log( AIR ) ~ P = 1  D = (1,12)  Q = (12)   NOINT
```

Figure 9.4. Model Selection from the HPFENGINE procedure

The following example shows how the HPFDIAGNOSE and HPFENGINE procedures can be used to select a single model specification from among multiple candidate model specifications.

In this example the HPFDIAGNOSE procedure creates three model specifications and adds them to the model repository SASUSER.MYCAT created in the previous example.

```
proc hpfdiag data=sashelp.air outest=est_three
     modelrepository=sasuser.mycat;
   id date interval=month;
   forecast air;
   arimax;
   esm;
   ucm;
run;


proc hpfengine data=sashelp.air print=(select)
     modelrepository=sasuser.mycat inest=est_three;
   forecast air;
   id date interval=month;
run;
```

If new model specification files are added to a model repository that already exists, then the suffixed number of the model specification file name and the model selection list file name are sequentially.

This example adds three model specification files, DIAG2, DIAG3, and DIAG4 to the model repository SASUSER.MYCAT which already contains DIAG0 and DIAG1.

Figure 9.5 shows the three model specifications (DIAG2, DIAG3, DIAG4) found by the HPFDIAGNOSE procedure.

```
                            The HPFENGINE Procedure

                                Model Selection
                               Criterion = RMSE

                        Model     Statistic    Selected

                        diag2     10.835333    No
                        diag3     10.652082    Yes
                        diag4     10.980119    No

                        Model Selection Criterion = RMSE

        Model    Label

        diag2    ARIMA: Log( AIR ) ~ P = 1  D = (1,12)  Q = (12)   NOINT
        diag3    Log Winters Method (Multiplicative)
        diag4    UCM: Log( AIR ) = TREND + SEASON + ERROR
```

Figure 9.5. Model Selection

Default Settings

The following example shows the HPFDIAGNOSE procedure with the default settings.

```
proc hpfdiag data=aaa print=all;
   id date interval=month;
   forecast y;
run;
```

It should be noted that the HPFDIAGNOSE procedure always performs the intermittency test first. If the HPFDIAGNOSE procedure determines that the series is intermittent, then the above example is equivalent to the following code:

```
proc hpfdiag data=aaa print=all;
   id date interval=month;
   forecast y;
   idm intermittent=2 base=auto;
run;
```

However, if the HPFDIAGNOSE procedure determines that the series is not intermittent, then the default settings are equivalent to the following code:

```
proc hpfdiag data=aaa print=all siglevel=0.05
     criterion=rmse holdout=0 holdoutpct=0 prefilter=yes
     back=0 errorcontrol=(severity=all stage=all);
   id date interval=month;
   forecast y;
   transform type=auto;
   trend dif=auto sdif=auto;
   arimax method=minic p=(0:5)(0:2) q=(0:5)(0:2) perror=(5:10)
```

```
      outlier=(detect=maybe maxnum=2 maxpct=2
                  siglevel=0.01 filter=full);
   esm method=best;
run;
```

The Role of the IDM Statement

The HPFDIAGNOSE procedure always performs the intermittency test first regardless of which model statement is specified. The IDM statement only controls the intermittency test using the INTERMITTENT= and BASE= options.

The following example specifies the IDM statement to control the intermittency test. If the HPFDIAGNOSE procedure determines that the series is intermittent, then an intermittent demand model is fitted to the data.

However, if the series is not intermittent, ARIMAX and ESM models are fitted to the data, even though the IDM statement is specified.

```
proc hpfdiag data=bbb print=all;
   id date interval=month;
   forecast x;
   idm intermittent=2.5 base=auto;
run;
```

The following example specifies the ESM statement. If the series is intermittent, an intermittent demand model is fitted to the data, even though the ESM statement is specified. But, if the series is not intermittent, an ESM model is fitted to the data. The same is true when the ARIMAX and UCM statements are specified.

```
proc hpfdiag data=ccc print=all;
   id date interval=month;
   forecast z;
   esm;
run;
```

Syntax

The HPFDIAGNOSE procedure uses the following statements:

PROC HPFDIAGNOSE *options*;
  BY *variables*;
  EVENT *event-names* ;
  FORECAST *variables* ;
  ID *variable* **INTERVAL=** *interval options*;
  INPUT *variables* ;
  TRANSFORM *options*;
  TREND *options*;
  ARIMAX *options*;

ESM *option*;
IDM *options*;
UCM *options*;
ADJUST *variable* **= (** *variable-list* **)** / *options*;

Functional Summary

The statements and options controlling the HPFDIAGNOSE procedure are summarized in the following table.

Description	Statement	Option
Statements		
specifies BY-group processing	BY	
specifies event definitions	EVENT	
specifies variables to be forecast	FORECAST	
specifies the time ID variable	ID	
specifies input variables	INPUT	
specifies log transform test and other functional transformation types	TRANSFORM	
specifies differencing test	TREND	
specifies ARIMAX model options	ARIMAX	
specifies exponential smoothing model	ESM	
specifies intermittent demand model options	IDM	
specifies unobserved components model	UCM	
specifies adjusting the dependent values	ADJUST	
Model Repository Options		
specifies the model repository	HPFDIAGNOSE	REPOSITORY=
specifies the base name for model specification files or model selection list files	HPFDIAGNOSE	BASENAME=
Data Set Options		
specifies the input data set	HPFDIAGNOSE	DATA=
specifies the mapping/estimate output data set	HPFDIAGNOSE	OUTEST=
specifies the events data set	HPFDIAGNOSE	INEVENT=
specifies the events data set organized by BY groups	HPFDIAGNOSE	EVENTBY=
specifies the output data set that contains the outliers	HPFDIAGNOSE	OUTOUTLIER=
Accumulation Options		
specifies length of seasonal cycle	HPFDIAGNOSE	SEASONALITY=
specifies accumulation frequency	ID	INTERVAL=
specifies interval alignment	ID	ALIGN=
specifies starting time ID value	ID	START=

Description	Statement	Option
specifies ending time ID value	ID	END=
specifies accumulation statistic	ID, FORECAST, INPUT, ADJUST	ACCUMULATE=
specifies missing value interpretation	ID, FORECAST, INPUT, ADJUST	SETMISSING=
specifies zero value interpretation	ID, FORECAST, INPUT, ADJUST	ZEROMISS=
specifies trim missing values	ID, FORECAST, INPUT, ADJUST	TRIMMISS=
Transformation Test Options		
specifies the AR order for the log transformation test	TRANSFORM	P=
specifies the type of the functional transformation	TRANSFORM	TYPE=
specifies the method of the forecasts of the transformed series	TRANSFORM	TRANSOPT=
Trend Test Options		
specifies the simple differencing	TREND	DIFF=
specifies the seasonal differencing	TREND	SDIFF=
specifies the AR order for the augmented unit root test	TREND	P=
ARIMAX Model Options		
specifies the ARMA order selection criterion	ARIMAX	CRITERION=
specifies the range of the AR orders for obtaining the error series used in the MINIC method	ARIMAX	PERROR=
specifies the range of the AR orders	ARIMAX	P=
specifies the range of the MA orders	ARIMAX	Q=
specifies the range of the denominator orders of the transfer function	ARIMAX	DEN=
specifies the range of the numerator orders of the transfer function	ARIMAX	NUM=
specifies the tentative order identification method	ARIMAX	METHOD=

Description	Statement	Option
specifies the outlier detection	ARIMAX	OUTLIER=
specifies the identification order of the components	ARIMAX	IDENTIFYORDER=
Unobserved Components Model Option		
specifies the components to test for inclusion in the UCM model	UCM	COMPONENT=
Exponential Smoothing Model Option		
specifies the method of the ESM model	ESM	METHOD=
Significance Level Option		
specifies the significance level for diagnostic tests	HPFDIAGNOSE, TRANSFORM, TREND, ARIMAX, UCM	SIGLEVEL=
specifies the significance level to control confidence limits in the model selection list files	HPFDIAGNOSE	ALPHA=
Event Variable Control Option		
specifies the maximum number of the events to be selected	HPFDIAGNOSE	SELECTEVENT=
specifies the required option of the event	EVENT	REQUIRED=
Input Variable Control Options		
specifies the maximum number of the input variables to be selected	HPFDIAGNOSE	SELECTINPUT=
specifies the transformation and differencing of the input variables	HPFDIAGNOSE	TESTINPUT=
specifies the required option of the variables	INPUT	REQUIRED=
Model Selection Options		
specifies the model selection criterion	HPFDIAGNOSE	CRITERION=
specifies the forecast holdout sample size	HPFDIAGNOSE	HOLDOUT=
specifies the forecast holdout sample percent	HPFDIAGNOSE	HOLDOUTPCT=
specifies data to hold back	HPFDIAGNOSE	BACK=
specifies the minimum number of observations needed to fit a trend or seasonal model	HPFDIAGNOSE	MINOBS=
Printing Options		

Description	Statement	Option
specifies printed output for only the model specifications	HPFDIAGNOSE	PRINT=SHORT
specifies printed output for PRINT=SHORT and summary of the transformation and trend tests	HPFDIAGNOSE	PRINT=LONG
specifies detailed printed output	HPFDIAGNOSE	PRINT=ALL
specifies control of message printing in the log	HPFDIAGNOSE	ERRORCONTROL=
Data Prefilter Option		
specifies handling missing and extreme values prior to diagnostic tests	HPFDIAGNOSE	PREFILTER=

PROC HPFDIAGNOSE Statement

PROC HPFDIAGNOSE *options ;*

The following options can be used in the PROC HPFDIAGNOSE statement:

ALPHA=*value*
specifies the confidence level size to use in computing the confidence limits in the model selection list files. The ALPHA= value must be between (0, 1). The default is ALPHA=0.05, which produces 95% confidence intervals.

BACK= *number*
specifies the number of observations before the end of the data. If BACK=n and the number of observation is T, then the first $T - n$ observations are used to diagnose a series. The default is BACK=0.

BASENAME= *SAS-name*
prefixes the model specification file name and/or the model selection list file name. If the BASENAME=MYSPEC, then the model specification files and/or the model selection list files are named MYSPEC0, ..., MYSPEC9999999999. The default SAS-name starts with DIAG, such as DIAG0, ..., DIAG9999999999. The model specification files and/or the model selection list files are stored in the model repository defined by the REPOSITORY= option.

CRITERION=*option*
specifies the model selection criterion to select the best model. This option would often be used in conjunction with the HOLDOUT= and HOLDOUTPCT= options. The default is CRITERION=RMSE. The following statistics of fit are provided:

SSE	Sum of Square Error
MSE	Mean Square Error
RMSE	Root Mean Square Error

UMSE	Unbiased Mean Square Error
URMSE	Unbiased Root Mean Square Error
MAXPE	Maximum Percent Error
MINPE	Minimum Percent Error
MPE	Mean Percent Error
MAPE	Mean Absolute Percent Error
MDAPE	Median Percent Error
GMAPE	Geometric Mean Percent Error
MINPPE	Minimum Predictive Percent Error
MAXPPE	Maximum Predictive Percent Error
MSPPE	Mean Predictive Percent Error
MAPPE	Symmetric Mean Absolute Predictive Percent Error
MDAPPE	Median Predictive Percent Error
GMAPPE	Geometric Mean Predictive Percent Error
MINSPE	Minimum Symmetric Percent Error
MAXSPE	Maximum Symmetric Percent Error
MSPE	Mean Symmetric Percent Error
SMAPE	Symmetric Mean Absolute Percent Error
MDASPE	Median Symmetric Percent Error
GMASPE	Geometric Mean Symmetric Percent Error
MINRE	Minimum Relative Error
MAXRE	Maximum Relative Error
MRE	Mean Relative Error
MRAE	Mean Relative Absolute Error
MDRAE	Median Relative Absolute Error
GMRAE	Geometric Mean Relative Absolute Error
MAXERR	Maximum Error
MINERR	Minimum Error
ME	Mean Error
MAE	Mean Absolute Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya’s Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
SBC	Schwarz Bayesian Information Criterion

APC	Amemiya's Prediction Criterion

DATA= *SAS data set*
specifies the name of the SAS data set containing the time series. If the DATA= option is not specified, the most recently created SAS data set is used.

HOLDOUT=*number*
specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of the dependent time series ending at the last nonmissing observation. The statistics of a model selection criterion are computed using only the holdout sample. The default is HOLDOUT=0.

HOLDOUTPCT=*value*
specifies the size of the holdout sample as a percentage of the length of the dependent time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where T is the length of the dependent time series with beginning and ending missing values removed. The default is HOLDOUTPCT=0.

INEVENT= *SAS data set*
specifies the name of the event data set containing the event definitions created by the HPFEVENTS procedure. If the INEVENT= data set is not specified, only SAS predefined event definitions can be used in the EVENT statement.

For more information on the INEVENT= option, see Chapter 12, "The HPFEVENTS Procedure."

EVENTBY= *SAS data set*
specifies the name of the event data set that contains the events for specific BY groups that are created by DATA steps. The events in the EVENT statement are used in all BY groups, but the events in the EVENTBY= data set are used in the specific BY group.

INSELECTNAME= *SAS-name*
specifies the name of a catalog entry that serves as a model selection list. This is the selection list that includes existing model specification files. A selection list created by the HPFDIAGNOSE procedure includes the existing model specification files.

ERRORCONTROL= (SEVERITY= (*severity-options* **) STAGE= (** *stage-options* **) MAXMESSAGE=** *number* **)**
allows finer control of message printing. The error severity level and the HPFDIAGNOSE procedure processing stages are set independently. The MAXMESSAGE=*number* option controls the number of messages printed. A logical 'and' is taken over all the specified options and any message.

Available *severity-options* are as follows:

LOW	specifies low severity, minor issues
MEDIUM	specifies medium severity problems
HIGH	specifies severe errors
ALL	specifies all severity levels of LOW, MEDIUM, and HIGH options

NONE	specifies that no messages from PROC HPFDIAGNOSE are printed

Available *stage-options* are as follows:

PROCEDURELEVEL	specifies that the procedure stage is option processing and validation
DATAPREP	specifies the accumulation of data and the application of SETMISS= and ZEROMISS= options
DIAGNOSE	specifies the diagnostic process
ALL	specifies all PROCEDURELEVEL, DATAPREP, and DIAGNOSE options

Examples are as follows:

```
errorcontrol=(severity=(high medium) stage=all);
```

prints high- and moderate-severity errors at any processing stage of PROC HPFDIAGNOSE.

```
errorcontrol=(severity=high stage=dataprep);
```

prints high-severity errors only during the data preparation.

```
errorcontrol=(severity=none stage=all);
errorcontrol=(maxmessage=0);
```

turns off messages from PROC HPFDIAGNOSE.

```
errorcontrol=( severity=(high medium low)
               stage=(procedurelevel dataprep diagnose) );
```

specifies the default behavior. Also the following code specifies the default behavior:

```
  errorcontrol=(severity=all stage=all)
```

MINOBS=(SEASON=*number* **TREND=***number***)**

SEASON=	specifies that no seasonal model is fitted to any series with fewer nonmissing observations than *number*$\times$ (season length). The value of *number* must be greater than or equal to 1. The default is *number* = 2.

TREND=	specifies that no trend model is fitted to any series with fewer non-missing observations than *number*. The value of *number* must be greater than or equal to 1. The default is *number* = 1.

OUTEST=*SAS data set*
contains information that maps data set variables to model symbols and references model specification files and model selection list files.

OUTOUTLIER=*SAS data set*
contains information associated with the detected outliers.

PREFILTER=*MISSING | YES | EXTREME | BOTH*
specifies handling missing and extreme values prior to diagnostic tests.

MISSING	Smoothed values for missing data are applied for tentative order selection and missing values are used for the final diagnostics.
YES	Smoothed values for missing data are applied to overall diagnoses. This option is the default.
EXTREME	Extreme values set to missing for a tentative ARIMA model and extreme values are used for the final ARIMAX model diagnostics.
BOTH	Both YES and EXTREME.

If the input variables have missing values, they are always smoothed for the diagnostics.

PRINT=*NONE | SHORT | LONG | ALL*
specifies the print option.

NONE	suppresses the printed output. This option is the default.
SHORT	prints the model specifications. This option also prints the only significant input variables, events, and outliers.
LONG	prints the summary of the transform, the stationarity test, and the determination of ARMA order including PRINT=SHORT.
ALL	prints the details of the stationarity test and the determination of ARMA order. This option prints the detail information about all input variables and events under consideration.

REPOSITORY= *catalog*
contains information about model specification files and model selection list files. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=. The default model repository is SASUSER.HPFDFLT.

SEASONALITY=*number*
specifies the length of the seasonal cycle. The *number* should be a positive integer. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. By default, the length of the seasonal cycle is 1 (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 12.

SELECTINPUT=*SELECT | ALL |number*

specifies the maximum number of the input variables to select.

SELECT	selects the input variables that satisfy the criteria (noncollinearity, nonnegative delay, smaller AIC). This option is the default.
ALL	selects the input variables that satisfy the criteria (noncollinearity, nonnegative delay).
number	selects the best *number* input variables that satisfy the criteria (noncollinearity, nonnegative delay).

SELECTEVENT=*SELECT | ALL |number*

specifies the maximum number of events to select.

SELECT	selects the events that satisfy the criteria (noncollinearity, smaller AIC). This option is the default.
ALL	selects the events that satisfy the criteria (noncollinearity).
number	selects the best *number* of events that satisfy the criteria (noncollinearity).

SIGLEVEL=*value*

specifies the cutoff value for all diagnostic tests such as log transformation, stationarity, tentative ARMA order selection, and significance of UCM components. The SIGLEVEL=value should be between (0,1) and SIGLEVEL=0.05 is the default. The SIGLEVEL options in TRANSFORM, TREND, ARIMAX, and UCM statements control testing independently.

TESTINPUT=*TRANSFORM | TREND |BOTH*

TRANSFORM	specifies that the log transform testing of the input variables is applied independently of the variable to be forecast.
TREND	specifies that the trend testing of the input variables is applied independently of the variable to be forecast.
BOTH	specifies that the log transform and trend testing of the input variables are applied independently of the variable to be forecast.

If the option is not specified, the same differencing is applied to the input variables as is used for the variable to be forecast, and no transformation is applied to the input variables.

BY Statement

BY *variables ;*

A BY statement can be used in the HPFDIAGNOSE procedure to process a data set in groups of observations defined by the BY variables.

ID Statement

ID *variable 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 FORECAST 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.

For more information on the ID statement, see the "ID Statement" section on page 363 in the HPFENGINE procedure.

ACCUMULATE=*option*
specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. The ACCUMULATE= option accepts the following values: NONE, TOTAL, AVERAGE|AVG, MINIMUM|MIN, MEDIAN|MED, MAXIMUM|MAX, N, NMISS, NOBS, FIRST, LAST, STDDEV|STD, CSS, USS. The default is NONE.

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" uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This option and the START= option can be used to ensure that data associated with each BY group contains the same number of observations.

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. 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. Refer to the *SAS/ETS User's Guide* for the intervals that can be specified.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. The SETMISSING= option accepts the following values: MISSING, AVERAGE|AVG, MINIMUM|MIN, MEDIAN|MED, MAXIMUM|MAX, FIRST, LAST, PREVIOUS|PREV, NEXT. The default is 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 END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contains the same number of observations.

TRIMMISS=*option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. The following options are provided:

NONE	No missing value trimming is applied.
LEFT	Beginning missing values are trimmed.
RIGHT	Ending missing values are trimmed.
BOTH	Both beginning and ending missing value are trimmed. This option is the default.

If the TRIMMISS= option is not specified in the FORECAST statement, missing values are set based on the TRIMMISS= option of the ID statement.

ZEROMISS=*option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. The following options are provided:

NONE	Beginning and/or ending zeros unchanged. This option is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

EVENT Statement

EVENT *event-names* **;**

The EVENT statement names event-names that identify the events in the INEVENT= data-set or predefined event-keywords or _ALL_.

The EVENT statement names either event-names or _ALL_. The event names identify the events in the INEVENT=data-set or are the SAS predefined event-keywords. _ALL_ is used to indicate that all simple events in the INEVENT=data set should be included in processing. If combination events exist in the INEVENT=data set and are to be included, then they must be specified in a separate EVENT statement. The HPFDIAGNOSE procedure does not currently process group events, although if the simple events associated with the group are defined in the INEVENT=data set, they can be included in processing, either by event-name or using _ALL_.

The EVENT statement requires the ID statement.

For more information on the EVENT statement, see Chapter 12, "The HPFEVENTS Procedure."

The following option can be used in the EVENT statement:

REQUIRED=*YES* **|** *MAYBE* **|** *NO*

The REQUIRED=YES specifies that the events are always included in the model as long as the model does not fail to be diagnosed.

The default is REQUIRED=NO.

The same differencing is applied to the events as is used for the variables to be forecast. No functional transformations are applied to the events.

FORECAST Statement

FORECAST *variables / options;*

Any number of FORECAST statements can be used in the HPFDIAGNOSE procedure. The FORECAST statement lists the variables in the DATA= data set to be diagnosed. The variables are dependent or response variables that you wish to forecast in the HPFENGINE procedure.

The following options can be used in the FORECAST statement:

ACCUMULATE=*option*

See the ACCUMULATE= option in the "ID Statement" section on page 305 for more details.

SETMISSING=*option* **|** *number*

See the SETMISSING= option in the "ID Statement" section on page 305 for more details.

TRIMMISS=*option*

See the TRIMMISS= option in the "ID Statement" section on page 305 for more details.

ZEROMISS=*option*

See the ZEROMISS= option in the "ID Statement" section on page 305 for more details.

INPUT Statement

INPUT *variables / options;*

Any number of INPUT statements can be used in the HPFDIAGNOSE procedure. The INPUT statement lists the variables in the DATA= data set to be diagnosed as regressors. The variables are independent or predictor variables to be used to forecast dependent or response variables.

The following options can be used in the INPUT statement:

REQUIRED=*YES* **|** *MAYBE* **|** *NO*

The REQUIRED=YES variables are always included in the model as long as the model does not fail to be diagnosed. The same differencing is applied to the REQUIRED=YES variables as is used for the variables to be forecast. No functional transformations are applied to the REQUIRED=YES variables. The delay and numerator and denominator orders of the REQUIRED=YES variables are set to zero.

The functional transform and differencing of the REQUIRED = MAYBE | NO variables depend on the request of the TESTINPUT option in the PROC HPFDIAGNOSE statement.

The default is REQUIRED=NO.

ACCUMULATE=*option*

See the ACCUMULATE= option in the "ID Statement" section on page 305 for more details.

SETMISSING=*option* **|** *number*

See the SETMISSING= option in the "ID Statement" section on page 305 for more details.

TRIMMISS=*option*

See the TRIMMISS= option in the "ID Statement" section on page 305 for more details.

ZEROMISS=*option*

See the ZEROMISS= option in the "ID Statement" section on page 305 for more details.

TRANSFORM Statement

TRANSFORM *<options>;*

A TRANSFORM statement can be used to specify the functional transformation of the series.

The following options can be used in the TRANSFORM statement:

P=*number*

specifies the autoregressive order for the log transform test. The default is P=$\min(2, [T/10])$ where T is the number of observations.

SIGLEVEL=*value*

specifies the significance level to use as a cutoff value to decide whether or not the series requires a log transformation. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

TRANSOPT=*MEAN | MEDIAN*

specifies whether mean or median forecasts are produced. If no transformation is applied to the series, then the mean and median forecasts are identical.

MEAN	The inverse transform produces mean forecasts. This is the default.
MEDIAN	The inverse transform produces median forecasts.

TYPE=*AUTO | LOG |NONE |SQRT | LOGISTIC |BOXCOX(value)*

specifies the type of functional transformation. The following transformations are provided:

AUTO	Automatically choose between NONE and LOG based on model selection criteria. If the TRANSFORM statement is not specified, this option is the default.
LOG	Logarithmic transformation. If the TYPE= option is not specified, this option is the default.
NONE	No transformation is applied.
SQRT	Square-root transformation.
LOGISTIC	Logistic transformation.
BOXCOX(*value*)	Box-Cox transformation with a parameter value where the value is between -5 and 5. The default is BOXCOX(1).

TREND Statement

TREND *options;*

A TREND statement can be used to test whether or not the dependent series requires simple or seasonal differencing, or both. The augmented Dickey-Fuller test (Dickey and Fuller 1979) is used for the simple unit root test.

If the seasonality is less than or equal to 12, the seasonal augmented Dickey-Fuller (ADF) test (Dickey, Hasza and Fuller 1984) is used for the seasonal unit root test. Otherwise, an AR(1) seasonal dummy test is used.

The joint simple and seasonal differencing test uses the Hasza-Fuller test (Hasza and Fuller 1979, 1984) in the special seasonality. Otherwise, proceed with the ADF test and the season dummy test.

The following options can be used in the TREND statement:

DIFF=*AUTO* **|** *NONE* **|***number* **| (0 :** *number***)**

AUTO	Tests for simple differencing. This option is the default.
NONE	Specifies that no simple differencing is to be used.
number	Specifies the simple differencing order. The option $number$=1 means $(1-B)y_t$ and $number$=2 means $(1-B)^2y_t$.
(0 : *number*)	Specifies the range of simple differencing order for testing. The option $number$ can be 0, 1, or 2.

SDIFF=*AUTO* **|** *NONE* **|***number*

AUTO	Tests for seasonal differencing. This option is the default.
NONE	Specifies the no seasonal differencing is to be used.
number	Specifies the seasonal differencing order. The option $number$=1 means $(1-B^s)y_t$ and $number$=2 means $(1-B^s)^2y_t$ where s is the seasonal period.

P=*number*
specifies the autoregressive order for the augmented unit root tests and a seasonality test. The default is P=$\min(5,[T/10])$ where T is the number of observations.

SIGLEVEL=*value*
specifies the significance level to use as a cutoff value to decide whether or not the series needs differencing. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

ARIMAX Statement

ARIMAX *<options>;*

An ARIMAX statement can be used to find an appropriate ARIMAX specification.

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the ARIMAX statement is not applicable. If the series is not intermittent, an ARIMAX model is fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and ESM models if the series is not intermittent, but diagnoses an IDM model if the series is intermittent.

The following options can be used in the ARIMAX statement:

PERROR=(*number* **:** *number***)**
specifies the range of the AR order for obtaining the error series used in the MINIC method. The default is (maxp:maxp+maxq).

P=(*number* **:** *number***) (***number* **:** *number***)**
specifies the range of the nonseasonal and seasonal AR orders. The default is (0:5)(0:2).

Q=(*number* **:** *number***) (***number* **:** *number***)**
specifies the range of the nonseasonal and seasonal MA orders. The default is (0:5)(0:2).

DEN=(*number* **:** *number***)**
specifies the range of the denominator order of the transfer function. The default is (0:2).

NUM=(*number* **:** *number***)**
specifies the range of the numerator order of the transfer function. The default is (0:2).

CRITERION=*AIC* **|** *SBC*
specifies the criterion for the tentative ARMA order selection. The default is CRITERION=SBC.

SIGLEVEL=*value*
specifies the significance level to use as a cutoff value to decide the AR and MA orders. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

ESTMETHOD=*CLS* **|** *ULS* **|** *ML*
specifies the method for choosing the tentative ARMA orders (Choi 1992; Tsay and Tiao 1984).

CLS	Conditional Least Squares method. This option is the default.
ULS	Unconditional Least Squares method.
ML	Maximum Likelihood method.

METHOD=*ESACF* **|** *MINIC* **|** *SCAN*
specifies the method for choosing the tentative ARMA orders (Choi 1992; Tsay and Tiao 1984).

ESACF	Extended Sample Autocorrelation Function.
MINIC	Minimum Information Criterion. This option is the default.
SCAN	Smallest Canonical Correlation Analysis.

OUTLIER=(*options***)**
specifies outlier detection in an ARIMAX model (de Jong and Penzer 1998).

DETECT=*YES*	includes outliers detected in a model if the model that includes the outliers is successfully diagnosed.
DETECT=*MAYBE*	includes outliers detected in a model if the model that includes the outliers is successfully diagnosed and has a smaller criterion than the model without outliers. This option is the default.

DETECT=*NO*	no outlier detection is performed.
FILTER=*FULL* | *SUBSET*	chooses a model for outlier detection. If FILTER=FULL, then use a full model. If FILTER=SUBSET, then use a subset model that includes nonseasonal AR and MA filters only. If the data have no seasonality, then the outlier detection is not affected by the FILTER= option. FILTER=FULL is the default.
MAXNUM=*number*	includes up to MAXNUM= value outliers in a model. MAXNUM=2 is the default.
MAXPCT=*value*	includes up to MAXPCT= value outliers in a model. MAXPCT=2 is the default. If MAXNUM=5 and MAXPCT=10, the number of the outliers is $min(5, 0.1T)$ where T is the length of the time series with beginning and ending missing values removed.
SIGLEVEL=*value*	specifies the cutoff value for outlier detection. The SIGLEVEL=value should be in (0,1). The SIGLEVEL=0.01 is the default. The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

If the OUTLIER= option is not specified, the HPFDIAGNOSE performs the outlier detection with the OUTLIER=(DETECT=MAYBE MAXNUM=2 MAXPCT=2 SIGLEVEL=0.01) option as default.

If the PREFILTER=EXTREME option is specified and extreme values are found, then these values are potential outliers. With the PREFILTER=EXTREME option, outliers may be detected even if the DETECT=NO option is specified; more than n number of outliers can be detected even if the MAXNUM=n option is specified.

IDENTIFYORDER= *ARIMA* | *REG* |*BOTH*
IDENTIFY= *ARIMA* | *REG* |*BOTH*

specifies the identification order when inputs and events are specified.

ARIMA	finds an ARIMA model for the error series first and then chooses significant inputs and events. This option is the default.
REG	finds a regression model first and then decides the AR and MA polynomial orders.
BOTH	fits models by using two methods and determines the better model.

ESM Statement

ESM *<option> ;*

An ESM statement can be used to find an appropriate ESM model specification based on the model selection criterion (McKenzie 1984).

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the ESM statement is not applicable. If the series is not intermittent, an ESM model is fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and ESM models if the series is not intermittent, but diagnoses an IDM model if the series is intermittent.

METHOD=*BEST|BESTN|BESTS*

BEST	fits the best candidate smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND, SEASONAL, WINTERS, ADDWINTERS). This is the default.
BESTN	fits the best candidate nonseasonal smoothing model (SIMPLE, DOUBLE, LINEAR, DAMPTREND).
BESTS	fits the best candidate seasonal smoothing model (SEASONAL, WINTERS, ADDWINTERS).

IDM Statement

IDM *<options> ;*

An IDM statement is used to control the intermittency test. The HPFDIAGNOSE procedure performs the intermittency test first.

If the series is intermittent, an intermittent demand model is fitted to the data based on the model selection criterion. However, if the series is not intermittent, ARIMAX and ESM models are fitted to the data.

If a model statement is not specified, the HPFDIAGNOSE procedure diagnoses ARIMAX and ESM models if the series is not intermittent, but diagnoses an IDM model if the series is intermittent.

INTERMITTENT=*number*
specifies a number greater than one that is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number then the series is assumed to be intermittent. The default is INTERMITTENT=2.

BASE=*AUTO | value*
specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's Method use BASE=0, which defines departures based on zero. The default is BASE=AUTO.

UCM Statement

UCM *<options> ;*

A UCM statement can be used to find an appropriate UCM model specification (Harvey 1989, 2001; Durbin and Koopman 2001).

The HPFDIAGNOSE procedure performs the intermittency test first. If the series is intermittent, an intermittent demand model is fitted to the data and the UCM statement is not applicable. If the series is not intermittent, a UCM model is fitted to the data.

The following options can be used in the UCM statement:

COMPONENT=(*components***)**

ALL	tests which components and/or variances are significant in the model. This option is the default. When the series has the seasonality information, the IRREGULAR, LEVEL, SLOPE, and SEASON components are included. Otherwise the IRREGULAR, LEVEL, SLOPE, and CYCLE components are included.
AUTOREG	tests if an *autoreg* component is significant in the model.
CYCLE	tests if two *cycle* components are significant in the model. The two CYCLE components are included and the LEVEL component is added. When the series has the seasonality information, the CYCLE component is not tested.
DEPLAG	tests if a *dependent lag* component is significant in the model. Only the order 1 is included.
IRREGULAR	tests if an *irregular* component is significant in the model.
LEVEL	tests if a *level* component is significant in the model.
SEASON	tests if a *season* component is significant in the model. When the series has the seasonality information, the SEASON component is not tested.
SLOPE	tests if a *slope* component is significant in the model. The LEVEL component is added.

SIGLEVEL=*value*

specifies the significance level to use as a cutoff value to decide which component and/or variances are significant. The SIGLEVEL=value should be in (0,1). The SIGLEVEL= option overrides the value of SIGLEVEL= option in the HPFDIAGNOSE statement.

ADJUST Statement

ADJUST *variable = (variable-list) / options***;**

The ADJUST statement lists the numeric variables in the DATA= data set whose accumulated values will be used to adjust the dependent values. Adjustments are performed before diagnostics.

The numeric variable listed is the variable to which adjustments specified in that statement will apply. This variable must appear in a FORECAST statement.

The numeric variables used as the source of the adjustments are listed following the parentheses. For more information see the "Adjustment Operations" section on page 316.

The following options can be used with the ADJUST statement:

OPERATION=*option*

specifies how the adjustments are applied to the forecast variable. The option determines how the adjustment variables are applied to the dependent variable prior to diagnostics.

Computations with missing values are handled differently in the ADJUST statement than in other parts of SAS. If any of the adjustment operations result in a nonmissing dependent value being added to, subtracted from, divided or multiplied by a missing value, the nonmissing dependent value is left unchanged. Division by zero produces a missing value.

The following predefined adjustment operations are provided:

NONE	No adjustment operation is performed. This is the default.
ADD	Variables listed in the adjustment statement are added to the dependent variable.
SUBTRACT	Variables listed in the adjustment statement are subtracted from the dependent variable.
MULTIPLY	Dependent variable is multiplied by variables listed in the adjustment statement.
DIVIDE	Dependent variable is divided by variables listed in the adjustment statement.
MIN	Dependent variable is set to the minimum of the dependent variable and all variables listed in the adjustment statement.
MAX	Dependent variable is set to the maximum of the dependent variable and all variables listed in the adjustment statement.

ACCUMULATE=*option*

See the ACCUMULATE= option in the "ID Statement" section on page 305 for more details.

SETMISSING=*option | number*

See the SETMISSING= option in the "ID Statement" section on page 305 for more details.

TRIMMISS=*option*

See the TRIMMISS= option in the "ID Statement" section on page 305 for more details.

ZEROMISS=*option*

See the ZEROMISS= option in the "ID Statement" section on page 305 for more details.

Details

Adjustment Operations

Pre-adjustment variables may be used to adjust the dependent series prior to model diagnostics.

If y_t is the dependent series and $z_{i,t}$ for $i = 1, \ldots, M$ are the M adjustment series, then the adjusted dependent series, w_t, is

$$\begin{aligned} w_t^1 &= op_i(y_t, z_{i,t}) \\ w_t^i &= op_i(w_t^{i-1}, z_{i,t}) \ \text{ for } \ 1 < i \leq M \\ w_t &= w_t^M \end{aligned}$$

where op_i represents the ith pre-adjustment operator and w_t^i is the ith adjusted dependent series. As can be seen, the pre-adjustment operators are nested and applied sequentially from $i = 1, \ldots, M$.

Data Preparation

The HPFDIAGNOSE procedure does not use missing data at the beginning and/or end of the series.

Missing values in the middle of the series to be forecast would be handled with the PREFILTER=MISSING or PREFILTER=YES option. The PREFILTER=MISSING option uses smoothed values for missing data for tentative order selection in the ARIMAX modeling and for tentative components selection in the UCM modeling, but the original values for the final diagnostics. The PREFILTER=YES option uses smoothed values for missing data and for all diagnostics.

Extreme values in the middle of the series to be forecast can be handled with the PREFILTER=EXTREME option in the ARIMA modeling. The HPFDIAGNOSE procedure replaces extreme values with missing values when determining a tentative ARIMA model, but the original values are used for the final diagnostics. The PREFILTER=EXTREME option detects extreme values if the absolute values of residuals are greater than 3 $\times$ STDDEV from a proper smoothed model.

If there are missing values in the middle of data for the input series, the procedure uses an interpolation method based on exponential smoothing to fill in the missing values.

The following data set provides a scenario for explaining the PREFILTER=EXTREME option.

```
data air_extreme;
   set sashelp.air;
   if _n_ = 30  then air = 500;
   if _n_ = 50  then air = 500;
   if _n_ = 100 then air = 700;
```

In the following SAS code, the HPFDIAGNOSE procedure diagnoses the new data set AIR_EXTREME without the PREFILTER=EXTREME option.

```
proc hpfdiag data=air_extreme print=short;
   id date interval=month;
   forecast air;
   arimax;
run;
```

In Figure 9.6, the ARIMA$(0, 1, 1)$ model is diagnosed for the time series. The model has no seasonality and is quite different from the model in Figure 9.1. The three extreme values mislead the model diagnostic tests.

```
                            The HPFDIAGNOSE Procedure

                            ARIMA Model Specification

         Functional                                              Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

AIR      NONE       NO        0  1  1  0  0  0          12 RMSE        67.1909

                            ARIMA Model Specification

                            Variable Status

                            AIR      OK
```

Figure 9.6. ARIMAX Model Specification without Outliers

In the following SAS code, the HPFDIAGNOSE procedure diagnoses the new data set AIR_EXTREME with the PREFILTER=EXTREME option.

```
proc hpfdiag data=air_extreme prefilter=extreme print=short;
   id date interval=month;
   forecast air;
   arimax;
run;
```

In Figure 9.7, the ARIMA$(1,1,0)(0,1,0)_{12}$ model is diagnosed for the time series. The required seasonal differencing is detected.

```
                              The HPFDIAGNOSE Procedure

                              ARIMA Model Specification

         Functional                                             Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

AIR      NONE       NO        1  1  0  0  1  0          12 RMSE        85.7080

                              ARIMA Model Specification

                              Variable Status

                              AIR      OK
```

Figure 9.7. ARIMAX Model Specification without Outliers

Figure 9.8 shows that the three extreme values are detected as outliers.

```
                               ARIMA Outlier Selection

                                                                  Approx
      Variable    Type        Obs        Time    Chi-Square    Pr > ChiSq

      AIR         AO          100     APR1957       1875.29        <.0001
                  AO           30     JUN1951       1820.42        <.0001
                  AO           50     FEB1953       1836.58        <.0001
```

Figure 9.8. Outlier Information

After the three extreme values are included in the ARIMAX model, Figure 9.8 shows that the statistic of the model selection criterion dramatically drops from 85.7080 to 11.5489.

```
               ARIMA Model Specification After Adjusting for Outliers

         Functional                                                     Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Outlier Criterion

AIR      NONE       NO        1  1  0  0  1  0          12       3 RMSE

                           ARIMA Model Specification After
                               Adjusting for Outliers

                       Variable Statistic    Status

                       AIR         11.5489    OK
```

Figure 9.9. ARIMAX Model Specification with Outliers

Functional Transformation

The log transform test compares the MSE or MAPE value after fitting an AR(p) model to the original data and to the logged data. If the MSE or MAPE value is smaller for the AR(p) model fitted to the logged data, then the HPFDIAGNOSE procedure will perform the log transformation.

The next two SAS programs specify the same log transformation test.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
run;
```

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
   transform type=auto;
run;
```

The Functional Transformation Table shown in Figure 9.10 states that the airline data requires a log transformation.

```
                         The HPFDIAGNOSE Procedure

                                Functional
                           Transformation Test

                                      Functional
                          Variable    Transform

                          AIR         LOG
```

Figure 9.10. Log Transformation Test

Stationarity Test

The stationarity test decides whether the data requires differencing. Note that d is the simple differencing order, and D is the seasonal differencing order.

The next two SAS programs specify the same trend test.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
run;
```

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
   trend diff=auto sdiff=auto;
run;
```

Simple Differencing Order

The simple augmented Dickey-Fuller test is used to determine the simple differencing order.

If there is no unit root, then the HPFDIAGNOSE procedure will set $d = 0$.

If there is a unit root, then the double unit root test is applied; if there is a double unit root, then the HPFDIAGNOSE procedure will set $d = 2$, otherwise $d = 1$.

Figure 9.11 and Figure 9.12 show that the series needs simple differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

```
                    Dickey-Fuller Unit Root Test

        Type                 Rho     Pr < Rho        Tau     Pr < Tau

        Zero Mean           0.22       0.7335       1.38       0.9580
        Single Mean        -2.42       0.7255      -1.11       0.7118
        Trend             294.41       0.9999      -6.42       <.0001
```

Figure 9.11. Dickey-Fuller Unit Root Test

```
                      Dickey-Fuller Unit Root Test Summary

                                          Zero
          Variable     Seasonality        Mean       Mean       Trend

          AIR                    1        YES        YES        NO
```

Figure 9.12. Summary of Dickey-Fuller Unit Root Test

Seasonal Differencing Order

The seasonal augmented Dickey-Fuller test is used to identify the seasonal differencing order. If the seasonality is greater than 12, the season dummy regression test is used. If there is no seasonal unit root, the HPFDIAGNOSE procedure will set $D = 0$. If there is a seasonal unit root, the HPFDIAGNOSE procedure will set $D = 1$.

Figure 9.13 and Figure 9.14 show that the series needs seasonal differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

```
          Seasonal Dickey-Fuller Unit Root Test(Seasonality=12)

        Type                 Rho     Pr < Rho         Tau     Pr < Tau

        Zero Mean          -0.47       0.5112       -0.13       0.4970
        Single Mean        -6.51       0.2186       -1.59       0.1101
```

Figure 9.13. Seasonal Dickey-Fuller Unit Root Test

```
              Seasonal Dickey-Fuller Unit Root Test Summary

                                       Zero
          Variable    Seasonality      Mean       Mean       Trend

          AIR                  12      YES        YES
```

Figure 9.14. Summary of Seasonal Dickey-Fuller Unit Root Test

Joint Differencing Orders

Hasza-Fuller (Hasza and Fuller 1979, 1984) proposed the joint unit roots test. If the seasonality is less than or equal to 12, use these tests. If there is a joint unit root, then the HPFDIAGNOSE procedure will set $D = 1$ and $d = 1$.

Figure 9.15 and Figure 9.16 show that the series needs both simple and seasonal differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

```
              Hasza-Fuller Joint Unit Root Test(Seasonality=12)

                                          Critical Values               Approx
   Type            F Value          90%          95%          99%       Pr > F

   Zero Mean        3.2684       2.5695       3.2600       4.8800       0.0466
   Single Mean      3.8360       5.1409       6.3473       8.8400       0.1476
   Trend            3.0426       7.2635       8.6820      10.7600       0.2896
```

Figure 9.15. Joint Unit Root Test

```
                        Joint Unit Root Test Summary

                                       Zero
          Variable    Seasonality      Mean       Mean       Trend

          AIR         1, 12            NO         YES
```

Figure 9.16. Summary of Joint Unit Root Test

Seasonal Dummy Test

If the seasonality is greater than 12, the seasonal dummy test is used to decide the seasonal differencing order.

The seasonal dummy test compares the criterion (AIC) of two AR(1) models and the joint significance of the seasonal dummy parameters, where one has seasonal dummy variables and the other does not have the seasonal dummy variables.

ARMA Order Selection

Tentative Simple Autoregressive and Moving-Average Orders

The tentative simple autoregressive and moving-average orders (AR=p^* and MA=q^*) are found using the ESACF, MINIC, or SCAN method.

The next two SAS programs result in the same diagnoses.

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax;
run;
```

```
proc hpfdiag data=sashelp.air print=all;
   id date interval=month;
   forecast air;
   arimax method=minic p=(0:5) q=(0:5) criterion=sbc;
run;
```

Figure 9.17 shows the minimum information criterion among the AR and MA orders. The AR=3 and MA=0 element has the smallest value in the table.

```
                    Minimum Information Criterion

       Lags       MA 0       MA 1       MA 2       MA 3       MA 4       MA 5

       AR 0   -6.20852   -6.30537   -6.29093    -6.3145   -6.28459   -6.26408
       AR 1   -6.31395   -6.28157   -6.26557   -6.28327   -6.25263   -6.23399
       AR 2   -6.29952   -6.26759   -6.24019   -6.24605   -6.21542   -6.20335
       AR 3   -6.33026   -6.29846   -6.26559   -6.23155    -6.2356   -6.22296
       AR 4   -6.31801   -6.28102   -6.24678   -6.24784   -6.21578   -6.19315
       AR 5   -6.29745    -6.2603   -6.22433    -6.2265   -6.19536   -6.15861
```

Figure 9.17. Minimum Information Criterion

Simple Autoregressive and Moving-Average Orders

The simple autoregressive and moving-average orders (p and q) are found by minimizing the SBC/AIC values from the models among $0 \leq p \leq p^*$ and $0 \leq q \leq q^*$ where p^* and q^* are the tentative simple autoregressive and moving-average orders.

Seasonal Autoregressive and Moving-Average Orders

The seasonal AR and MA orders (P and Q) are found by minimizing the SBC/AIC values from the models among $0 \leq P \leq 2$ and $0 \leq Q \leq 2$.

Constant

In order to determine whether the model has a constant, two models are fitted: $(p, d, q)(P, D, Q)_s$ and $C + (p, d, q)(P, D, Q)_s$. The model with the smaller SBC/AIC value is chosen.

Estimation Method

The ARIMA model uses the conditional least-squares estimates for the parameters.

Figure 9.18 shows that the simple AR and MA orders are reduced to $p = 1$ and $q = 0$ from $p^* = 3$ and $q^* = 0$. The seasonal AR and MA orders are $P = 0$ and $Q = 1$. The selected model does not have a constant term.

```
                              ARIMA Model Specification

         Functional                                              Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

AIR      LOG        NO        1  1  0  0  1  1          12 RMSE        10.8353

                              ARIMA Model Specification

                              Variable Status

                              AIR      OK
```

Figure 9.18. ARIMAX Specification

Transfer Functions in an ARIMAX Model

A transfer function filter has delay, numerator, and denominator parameters. Set (b, k, r) where b is the delay, k is the numerator order, and r is the denominator order.

Functional Transformation for Input Variables

The default of functional transformation for the inputs is no transformation. The TESTINPUT=TRANSFORM option specifies that the same functional transformation is applied to the inputs as is used for the variable to be forecast.

Using the TESTINPUT=TRANSFORM option, you can test whether the log transformation is applied to the inputs.

Simple and Seasonal Differencing Orders for Input Variables

The default of the simple and seasonal differencing for the inputs is the same as the simple and seasonal differencing applied to the variable to be forecast.

Using the TESTINPUT=TREND option, you can test whether the differencing is applied to the inputs.

Cross-Correlations between Forecast and Input Variables

The cross-correlations between the variable (y_t) to be forecast and each input variable (x_{it}) are used to identify the delay parameters. The following steps are used to prewhiten the variable to be forecast in order to identify the delay parameter (b).

1. Find an appropriate ARIMA model for x_{it} and estimate the residual of x_{it} (e^x_{it}).
2. Prewhiten y_t using this model and get the residual of y_t (e^y_{it}).
3. Compute the cross-correlations between e^x_{it} and e^y_{it} and find the first significant lag that is zero or larger. If no delay lag is significant, the variable x_{it} is not included in the model.

Simple Numerator and Denominator Orders

The high-order lag regression model and the transfer function model are compared to identify the simple numerator and denominator orders.

Fit the high-order lag regression model (lag=15) and get the coefficients. Fit the transfer function $C + (b, k, r)$ where C is a constant term, b is the delay parameter found in the previous section, $0 \leq k \leq 2$, and $0 \leq r \leq 2$, and get the impulse weight function (lag=15) of the transfer model. Compare the pattern of the coefficients from the high-order regression model and the transfer model.

The following SAS code shows how to select significant input variables.

```
proc hpfdiag data=sashelp.citimon(obs=141) print=all;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
   arimax;
run;
```

The ARIMA Input Selection Table shown in Figure 9.19 states that the EEGP input variable is selected in the model with differences $d = 2$, delay=8, and denominator order=2. Other input variables are not selected because of either unstable or insignificant status.

```
                                The HPFDIAGNOSE Procedure

                                  ARIMA Input Selection

Input              Functional
Variable  Selected Transform   d Delay Numerator Denominator Status

CCIUTC    NO       NONE        2     2         1           1 UnStable
EEC       NO       NONE        2     2         0           2 UnStable
EEGP      YES      NONE        2     8         0           2 OK
EXVUS     NO                                                 Not Significant
FM1       NO                                                 Not Significant
FM1D82    NO                                                 Not Significant
```

Figure 9.19. ARIMA Input Selection

Outliers

Outlier detection is the default in the ARIMAX modeling.

There are two types of outliers: the additive outlier (AO) and the level shift (LS). For each detected outlier, dummy regressors or indicator variables are created. The ARIMAX model and the dummy regressors are fitted to the data.

The detection of outliers follows a forward method. First find a significant outlier. If there are no other significant outliers, detecting outlier stops at this point. Otherwise, include this outlier into a model as an input and find another significant outlier.

The same functional differencing is applied to the outlier dummy regressors as is used for the variable to be forecast.

The data came from Hillmer, Larcker, and Schroeder (1983).

```
data hardware;
   input hardware @@;
   label hardware="Wholesale Sales of Hardware";
   date=intnx('month','01jan67'd,_n_-1);
   format date monyy.;
datalines;
 626  614  689  686  723  778  711  824  793  831  775  689
 692  718  757  769  791  809  836  878  856  935  850  763
 761  796  830  902  910  932  931  908  934  995  865  822
 763  778  841  845  863  952  909  899  952  963  893  831
 773  803  918  967  963 1065 1014 1051 1054 1051 1039  960
 930  956 1072 1023 1136 1181 1088 1247 1164 1251 1218 1062
1114 1088 1253 1254 1354 1349 1305 1420 1313 1481 1387 1284
1310 1262 1446 1573 1634 1612 1591 1640 1590 1696 1456 1296
1311 1232 1274 1388 1374 1443 1466 1454 1538 1587 1406 1341
1351 1367 1553 1588 1591 1703 1643 1711 1731 1678 1678 1580
1515 1544 1817 1838 1925 2017 1898 2068 1961 2027 1974 1820
1790 1708 2021 2102 2306 2360 2247 2412 2159 2455 2250 2057
2142 1984 2319 2374 2592 2461 2524 2678 2399 2794 2415
;
```

The next two SAS programs result in the same outlier analysis.

```
proc hpfdiag data=hardware print=short;
   id date interval=month;
   forecast hardware;
   arimax;
run;
```

```
proc hpfdiag data=hardware print=short;
   id date interval=month;
   forecast hardware;
   arimax outlier=(detect=maybe maxnum=2 maxpct=2 siglevel=0.01);
run;
```

Figure 9.20 shows that the two level shifts (LS) occurred at the 96th (DEC1974) and 99th (MAR1975) observations.

```
                              The HPFDIAGNOSE Procedure

                               ARIMA Outlier Selection

                                                                         Approx
       Variable     Type         Obs        Time     Chi-Square     Pr > ChiSq

       hardware     LS            99     MAR1975          25.73         <.0001
                    LS            96     DEC1974          29.64         <.0001
```

Figure 9.20. Outlier Information

Figure 9.21 shows the ARIMA model specification with two outliers included in the model.

```
              ARIMA Model Specification After Adjusting for Outliers

          Functional                                              Model
  Variable Transform  Constant  p  d  q  P  D  Q Seasonality Outlier Criterion

  hardware NONE        NO       2  1  1  2  1  1          12       2 RMSE

                        ARIMA Model Specification After
                            Adjusting for Outliers

                    Variable Statistic    Status

                    hardware   45.9477    OK
```

Figure 9.21. ARIMAX Specification

OUTOUTLIER

The OUTOUTLIER= data set contains information about the outliers and has the following variables:

BY variable name	Contains BY variables that organize the results in BY groups.
NAME	Contains variable(s) to be forecast.
TYPE	Contains a type, either AO for an *additive* outlier or LS for a *level shift*.
DIRECTION	Contains a direction, either UP for a positive effect or DOWN for a negative effect.
OBS	Contains the number of the observation where the outlier happens.
SASDATE	Contains the SAS date when the outlier happens.
EVENT	Contains the outlier's event name.
ESTIMATE	Contains the coefficient estimate.

CHISQ	Contains the chi-square statistic of the coefficient estimate.
PVALUE	Contains the *p*-value of the coefficient estimate.

The following SAS code and Figure 9.22 show the OUTOUTLIER= data set that contains information associated with the output in Figure 9.20.

```
proc hpfdiag data=hardware outoutlier=outl;
   id date interval=month;
   forecast hardware;
   arimax;
run;

proc print data=outl;
```

```
                    _
                    D                             _
                    I                             E
                    R          _                  S
                    E          A                  T            _
                    C          S          _       I     _      P
     _      _       T   _      D          E       M     C      V
     N      T       I   O      A          V       A     H      A
O    A      Y       O   B      T          E       T     I      L
b    M      P       N   S      E          N       E     S      U
s    E      E       _   _      _          T       _     Q      E
     _      _                             _             _      _

1  hardware LS   DOWN  99  01MAR75  LS01MAR1975D  -125.695  25.7273  .000000393
2           LS   DOWN  96  01DEC74  LS01DEC1974D  -115.714  29.6352  .000000052
```

Figure 9.22. OUTOUTLIER Data Set

Intermittent Demand Model

The HPFDIAGNOSE procedure selects an appropriate intermittent demand model (IDM) based on the model selection criterion.

If a series is intermittent or interrupted, a proper IDM is selected by either individually modeling both the demand interval and size component or jointly modeling these components using the average demand component (demand size divided by demand interval).

The following example prints the diagnostics of an intermittent demand series. The INTERMITTENT=2.5 and BASE=0 are specified.

```
data sales;
   input hubcaps @@;
datalines;
0 1 0 0 0 1 0 0 0 0 0 2 0 4 0 0 0 0 1 0
;
```

```
proc hpfdiag data=sales print=all;
   forecast hubcaps;
   idm intermittent=2.5 base=0;
run;
```

Figure 9.23 shows that the variable to be forecast is an intermittent demand series. The Interval/Size demand model and Average demand model were diagnosed for the time series. The value of the model selection criterion of the Average demand model is smaller than that of the Interval/Size demand model.

```
                               The HPFDIAGNOSE Procedure

                        Intermittent Demand Model Specification

          Demand     Functional  Selected                 Model
Variable  Model      Transform   Model        Component   Criterion  Statistic

hubcaps   INTERVAL   NONE        DOUBLE       LEVEL       RMSE          0.8288
          SIZE       LOG         SIMPLE       LEVEL
          AVERAGE    LOG         SIMPLE       LEVEL                     0.8736
```

Figure 9.23. Intermittent Demand Model Specification

Exponential Smoothing Model

The HPFDIAGNOSE procedure selects an appropriate exponential smoothing model (ESM) based on the model selection criterion.

The following example prints the ESM model specification.

```
proc hpfdiag data=sashelp.gnp print=short;
   id date interval=qtr;
   forecast gnp;
   esm;
run;
```

The ESM model specification in Figure 9.24 states that the damp-trend exponential smoothed model was automatically selected.

```
                               The HPFDIAGNOSE Procedure

                         Exponential Smoothing Model Specification

            Functional    Selected                     Model
Variable    Transform     Model          Component     Criterion    Statistic

GNP         NONE          DAMPTREND      LEVEL         RMSE           22.0750
                                         TREND
                                         DAMP
```

Figure 9.24. ESM Specification

Unobserved Components Model

The UCM statement is used to find the proper components among the level, trend, seasonal, cycles, and regression effects.

Differencing Variables in a UCM

The variable to be forecast and the events are not differenced regardless of the result of the TREND statement. Differencing of the input variables follows the result of the option TESTINPUT=TREND or TESTINPUT=BOTH.

Transfer Function in a UCM

The functional transformation, simple and seasonal differencing, and delay parameters for the transfer function in a UCM are the same as those that are used for the transfer function in an ARIMAX model.

The series that consists of the yearly river flow readings of the Nile, recorded at Aswan (Cobb 1978), is studied. The data consists of readings from the years 1871 to 1970.

The following DATA step statements read the data in a SAS data set and create dummy inputs for the shift in 1899 and the unusual years 1877 and 1913.

```
data nile;
   input riverFlow @@;
   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
;
```

The series is known to have had a shift in the level starting at the year 1899, and the years 1877 and 1913 are suspected to be outlying points. The following SAS code creates the NILE_DATA data set with the Shift1899, Event1877, and Event1913 variables.

```
data nile_data;
   set nile;
   if year >= '1jan1899'd then Shift1899 = 1.0;
   else Shift1899 = 0;
   if year = '1jan1913'd then Event1913 = 1.0;
   else Event1913 = 0;
```

```
      if year = '1jan1877'd then Event1877 = 1.0;
      else Event1877 = 0;
```

The following SAS codes prints the diagnoses of the UCM model specification.

```
   proc hpfdiag data=nile_data print=short;
      id year interval=year;
      forecast riverFlow;
      input Shift1899 Event1913 Event1877;
      ucm;
   run;
```

Figure 9.25 shows the three significant inputs chosen.

```
                          The HPFDIAGNOSE Procedure

                             UCM Input Selection

              Input                       Functional
              Variable     Selected       Transform      d     Delay

              Shift1899    YES            NONE           0         0
              Event1913    YES            NONE           0         0
```

Figure 9.25. UCM Input Selection

Figure 9.26 shows the UCM model specification for the Nile data. The data has a significant cycle, level components, and the two inputs.

```
                  Unobserved Components Model(UCM) Specification

          Functional                                             Model
Variable  Transform  Component  Selected Stochastic      Period Criterion Statistic

riverFlow NONE       IRREGULAR  NO                             RMSE         116.67
                     LEVEL      YES      NO
                     SLOPE      NO
                     CYCLE1     YES      YES            4.1112
                     CYCLE2     NO
                     INPUT      2

                             Unobserved Components
                           Model(UCM) Specification

                          Variable  Status

                          riverFlow OK
```

Figure 9.26. UCM Specification

The following example has the same results as Figure 9.25. The COMPONENTS= option in the UCM statement specifies level and cycle as components to consider.

```
proc hpfdiag data=nile_data print=short;
   id year interval=year;
   forecast riverFlow;
   input Shift1899 Event1913 Event1877;
   ucm component=(level cycle);
run;
```

Holdout Sample

A holdout sample is useful to find models that have better out-of-sample forecasts. If the HOLDOUT= or HOLDOUTPCT= option is specified, the model selection criterion is computed using only the holdout sample region.

```
proc hpfdiag data=sashelp.air print=short holdout=10;
   id date interval=month;
   forecast air;
   arimax;
run;
```

The ARIMA model specification in Figure 9.27 shows that the log test, trend test, and selection of ARMA orders use only the first part of the series and exclude the last 10 observations that were specified as the holdout sample. The statistic of the model selection criterion is computed using only the last 10 observations that were specified as the holdout sample.

```
                                 The HPFDIAGNOSE Procedure

                                 ARIMA Model Specification

         Functional                                              Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

AIR      LOG        NO        1  1  0  0  1  1          12 RMSE        16.3008

                                 ARIMA Model Specification

                                    HoldOut
                          Variable   Sample    Status

                          AIR            10    OK
```

Figure 9.27. Use HOLDOUT Option

EVENTS

Calendar effects such as holiday and trading day are defined by the HPFEVENTS procedure or predefined event-keywords.

The HPEVENTS procedure creates the OUT= data set for the event definitions, and the HPFDIAGNOSE procedure uses these event definitions by specifying the INEVENT= option in the ARIMAX or UCM model.

Events in an ARIMAX Model

The simple and seasonal differencing for the events in an ARIMAX are the same as those that are used for the variable to be forecast.

No functional transformations are applied to the events.

Events in a UCM

The simple and seasonal differencing for the events in a UCM model are not applied to the events.

No functional transformations are applied to the events.

The following SAS code shows how the HPEVENTS procedure can be used to create the event data set, OUT=EVENTDATA.

```
proc hpfevents data=nile;
   id year interval=year;
   eventkey Shift1899 = LS01JAN1899D;
   eventkey Event1913 = AO01JAN1913D;
   eventkey Event1877 = AO01JAN1877D;
   eventdata out=eventdata;
run;
```

The following SAS code shows that the HPFDIAGNOSE procedure uses this event data by specifying the INEVENT=EVENTDATA option. The EVENT statement specifies the name of events defined in the INEVENT=EVENTDATA.

```
proc hpfdiag data=nile print=short inevent=eventdata;
   id year interval=year;
   forecast riverFlow;
   event Shift1899 Event1913 Event1877;
   ucm component=(level cycle);
run;
```

Figure 9.28 shows the three significant events chosen.

```
                              The HPFDIAGNOSE Procedure

                                 UCM Event Selection

                                Event
                                Name          Selected

                                SHIFT1899     YES
                                EVENT1913     YES
```

Figure 9.28. UCM Event Selection

Figure 9.29 shows the UCM model specification for the Nile data. The data has the significant cycle, level components, and the two events.

```
                 Unobserved Components Model(UCM) Specification

          Functional                                              Model
Variable  Transform   Component  Selected Stochastic   Period Criterion Statistic

riverFlow NONE        LEVEL      YES      NO                  RMSE         116.67
                      CYCLE1     YES      YES          4.1112
                      CYCLE2     NO
                      EVENT      2

                                    Unobserved Components
                                  Model(UCM) Specification

                                 Variable  Status

                                 riverFlow OK
```

Figure 9.29. UCM Specification

The following program generates the same results as the previous example without specifying an INEVENT= data set. In this example, SAS predefined event-keywords are specified in the EVENT statement.

```
proc hpfdiag data=nile print=short;
   id year interval=year;
   forecast riverFlow;
   event LS01JAN1899D AO01JAN1913D AO01JAN1877D;
   ucm component=(level cycle);
run;
```

HPFENGINE

The HPFDIAGNOSE procedure diagnoses and the HPFENGINE procedure forecasts.

There are two ways to communicate between the HPFDIAGNOSE procedure and the HPFENGINE procedure. One way is that the OUTEST= data set specified in the HPFDIAGNOSE procedure is specified as the INEST= data set in the HPFENGINE procedure. The other way is that the HPFSELECT procedure is used to communicate between the HPFDIAGNOSE procedure and the HPFENGINE procedure.

The ALPHA=, CRITERION=, HOLDOUT=, and HOLDOUTPCT= options can be changed using the HPFSELECT procedure before these options are transmitted to the HPFENGINE procedure. Otherwise the values specified in the HPFDIAGNOSE procedure are transmitted directly to the HPFENGINE procedure.

Missing values in the input series are handled differently in the HPFDIAGNOSE procedure than in the HPFENGINE procedure. The HPFDIAGNOSE procedure uses the smoothed missing values for inputs, but the HPFENGINE procedure does not include the inputs that have missing values. This difference can produce different statistical results between the two procedures.

The model specification files created by the HPFDIAGNOSE procedure can be compared with benchmark model specifications using the HPFESMSPEC, HPFIDMSPEC, HPFARIMASPEC, and HPFUCMSPEC procedures.

The following example shows how to combine these procedures to diagnose a time series.

Create a diagnosed model specification.

```
proc hpfdiag data=sashelp.air outest=est
   modelrepository=sasuser.mymodel;
   id date interval=month;
   forecast air;
   arimax;
run;
```

Create an ARIMA$(0,1,1)(0,1,1)_s$ model specification.

```
proc hpfarimaspec modelrepository=sasuser.mymodel
     specname=benchModel;
   forecast var=dep1 dif=1 12 q=(1)(12) noint transform=log;
run;
```

Create a model selection list that includes a diagnosed model (DIAG0) and a specified model (BENCHMODEL).

```
proc hpfselect modelrepository=sasuser.mymodel
     selectname=arimaSpec;
   select criterion=mape;
   spec diag0 / eventmap(symbol=_none_ event=ao135obs)
                eventmap(symbol=_none_ event=ao29obs);
   spec benchModel / inputmap(symbol=dep1 data=air);
run;
```

Select a better model from the model specification list.

```
proc hpfengine data=sashelp.air print=(select)
     modelrepository=sasuser.mymodel
     globalselection=arimaSpec;
   forecast air;
   id date interval=month;
run;
```

Figure 9.30 shows the DIAG0 and BENCHModel model specifications. The DIAG0.XML is created by the HPFDIAGNOSE procedure and the BENCHModel is created by the HPFARIMASPEC procedure.

```
                          The HPFENGINE Procedure

                      Model Selection Criterion = MAPE

                    Model          Statistic    Selected

                    DIAG0          2.7094770    Yes
                    BENCHMODEL     2.8979097    No

                     Model Selection Criterion = MAPE

     Model         Label

     DIAG0         ARIMA: Log( AIR ) ~ P = 1  D = (1,12)  Q = (12)   NOINT
     BENCHMODEL    ARIMA: Log( DEP1 ) ~ D = (1,12)  Q = ((1)(12))   NOINT
```

Figure 9.30. Model Selection from the HPFENGINE Procedure

OUTEST

The OUTEST= data set contains information that maps data set variables to model symbols and references the model specification file and model selection list files for each variable to be forecast. This information is used by the HPFENGINE procedure for further model selection, parameter estimation, and forecasts.

In addition, this information can be used by the HPFSELECT procedure to create customized model specification files.

The OUTEST= data set has the following columns;

BY variable name	Contains BY variables that organize the results in BY groups.
NAME	Contains variable(s) to be forecast.
SELECT	Contains model selection list file names. The model selection list file contains the information of the values of CRITERION=, ALPHA=, HOLDOUT=, and HOLDPCT= options, EVENT and OUTLIER information, and model specification file names.
MODEL	Not applicable in the HPFDIAGNOSE procedure.
SCORE	Not applicable in the HPFDIAGNOSE procedure.
MODELVAR	Model symbol.
DSVAR	Data set variable name.
VARTYPE	DEPENDENT.

Here are two examples. The first has one model specification file with a model selection list file; the second one has two model select list files and four model specification files.

The first example uses the BASENAME=AIRSPEC and the new model repository SASUSER.MYMODEL.

```
proc hpfdiag data=sashelp.air outest=est_air
   modelrepository=sasuser.mymodel basename=airSpec;
   id date interval=month;
   forecast air;
   arimax;
run;

proc print data=est_air;
run;
```

Figure 9.31 shows _SELECT_=AIRSPEC1 since BASENAME=AIRSPEC is specified. Because the new model repository SASUSER.MYMODEL is created, the suffix number followed by AIRSPEC starts from 0. AIRSPEC0 is the model specification file and AIRSPEC1 is the model selection list file.

```
Obs   _NAME_   _SELECT_   _MODEL_   _SCORE_   _MODELVAR_   _DSVAR_   _VARTYPE_

 1     AIR     AIRSPEC1                            Y          AIR     DEPENDENT
```

Figure 9.31. OUTEST1

The next example uses the new BASENAME=GNPSPEC and the new model repository SASUSER.MYGNP. The ESM and ARIMAX statement are requested for two variables to be forecast.

```
proc hpfdiag data=sashelp.gnp outest=est_gnp
   modelrepository=sasuser.myGNP basename=gnpSpec;
   id date interval=qtr;
   forecast consump invest;
   esm;
   arimax;
run;

proc print data=est_gnp;
```

Figure 9.32 shows two observations. Since the model repository SASUSER.MYGNP is newly created, the suffix number followed by GNPSPEC starts from 0.

The model selection list GNPSPEC2 contains the two model specifications; GNPSPEC0 is the ARIMAX model specification, and GNPSPEC1 is the ESM model specification for the variable to be forecast, CONSUMP.

The model selection list GNPSPEC5 contains the two model specifications; GNPSPEC3 is the ARIMAX model specification, and GNPSPEC4 is the ESM model specification for the variable to be forecast, INVEST.

```
Obs    _NAME_     _SELECT_    _MODEL_    _SCORE_    _MODELVAR_    _DSVAR_    _VARTYPE_

 1     CONSUMP    GNPSPEC2                               Y        CONSUMP    DEPENDENT
 2     INVEST     GNPSPEC5                               Y        INVEST     DEPENDENT
```

Figure 9.32. OUTEST2

ODS Table Names

The HPFDIAGNOSE 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 Table 9.1:

Table 9.1. ODS Tables Produced Using the HPFDIAGNOSE Procedure

ODS Table Name	Description	Statement	Option
ODS Tables Created by the PRINT=SHORT Option			
ARIMAEventSelect	ARIMA Event Selection	EVENT	
ARIMAInputSelect	ARIMA Input Selection	INPUT	
ARIMASpec	ARIMA Model Specification	ARIMAX	
BestModelSpec	Selected Model Specification		
ESMSpec	Exponential Smoothing Model Specification	ESM	
FinalARIMASpec	Final ARIMA Model Specification	ARIMAX	
IDMSpec	Intermittent Model Specification	IDM	
OutlierInfo	ARIMA Outlier Selection	ARIMAX	
UCMEventSelect	UCM Event Selection	EVENT	
UCMInputSelect	UCM Input Selection	INPUT	
UCMSpec	Unobserved Components Model Specification	UCM	
VariableInfo	Forecast Variable Information		
Additional ODS Tables Created by the PRINT=LONG Option			
DFTestSummary	Dickey-Fuller Unit Root Test	TREND	DIFF
JointTestSummary	Joint Unit Root Test	TREND	DIFF, SDIFF
SeasonDFTestSummary	Seasonal Dickey-Fuller Unit Root Test	TREND	SDIFF
Transform	Functional Transformation Test	TRANSFORM	TYPE=AUTO
Additional ODS Tables Created by the PRINT=ALL Option			
DFTest	Dickey-Fuller Unit Root Test	TREND	DIFF

Table 9.1. (ODS Tables Continued)

ODS Table Name	Description	Statement	Option
ESACF	Extended Sample Autocorrelation Function	ARIMAX	ESACF
ESACFPValues	P-values of ESACF	ARIMAX	ESACF
JointTest	Joint Unit Root Test	TREND	DIFF, SDIFF
MINIC	Minimum Information Criterion	ARIMAX	MINIC
SCAN	Squared Canonical Correlation Estimates	ARIMAX	SCAN
SCANPValues	P-values of SCAN	ARIMAX	SCAN
SeasonDFTest	Seasonal Dickey-Fuller Unit Root Test	TREND	SDIFF

Examples

Example 9.1. Selection of Input Variables

This example requests testing of the transformation and differencing of the input variables independent of the variable to be forecast.

```
proc hpfdiag data=sashelp.citimon(obs=141)
      testinput=both selectinput=all print=all;
   forecast conb;
   input cciutc eec eegp exvus fm1 fm1d82;
   arimax;
run;
```

Output 9.1.1 shows that the ARIMA $(0, 2, 1)$ model is diagnosed for the variable (CONB) to be forecast.

Output 9.1.1. ARIMAX Specification before Input Selection

```
                            The HPFDIAGNOSE Procedure

                           ARIMA Model Specification

         Functional                        Model
Variable Transform  Constant  p  d  q Criterion Statistic Status

CONB     NONE       NO        0  2  1 RMSE        2318.33 OK
```

Output 9.1.2 shows that one input variable (EEGP) is selected. The input variable needs a simple differencing.

Output 9.1.2. ARIMA Input Selection

```
                              The HPFDIAGNOSE Procedure

                                ARIMA Input Selection

 Input              Functional
 Variable Selected  Transform   d Delay Numerator Denominator Status

 CCIUTC   NO        NONE        1     4         0           0 UnStable
 EEC      NO                                                  Not Significant
 EEGP     YES       NONE        1     9         2           2 OK
 EXVUS    NO                                                  Not Significant
 FM1      NO                                                  Not Significant
 FM1D82   NO                                                  Not Significant
```

Output 9.1.3 shows the outlier detection information. The 136th observation is detected as a significant level shift (LS); the 120th observation is detected as a significant additive outlier (AO).

Output 9.1.3. Outlier Detection

```
                               ARIMA Outlier Selection

                                                            Approx
            Variable     Type         Obs     Chi-Square    Pr > ChiSq

            CONB         LS           136          26.50        <.0001
                         AO           120          16.76        <.0001
```

Output 9.1.4 shows that the RMSE model selection criterion with inputs is smaller than the model selection criterion without inputs and outliers.

Output 9.1.4. ARIMAX Specification after Input Selection

```
        ARIMA Model Specification After Adjusting for Inputs and Outliers

           Functional                                     Model
Variable   Transform    Constant   p   d   q  Outlier  Input  Criterion  Statistic

CONB       NONE         NO         0   2   1        2      1  RMSE         2021.66

                             ARIMA Model Specification
                                After Adjusting for
                                Inputs and Outliers

                            Variable   Status

                            CONB       OK
```

Example 9.2. Selection of Events and Input Variables

This example demonstrates how to select events and input variables.

```
proc hpfevents data=sashelp.gnp;
   id date interval=qtr;
   eventkey shock=AO105OBS;
   eventkey shift=LS85OBS;
   eventdata out=eventdata;
run;


proc hpfdiag data=sashelp.gnp print=all inevent=eventdata
     testinput=trend;
   id date interval=qtr;
   forecast gnp;
   input consump invest exports govt;
   event shock shift;
   arimax outlier=(detect=no);
run;
```

Output 9.2.1 shows the seasonal ARIMA $(0, 2, 1)(2, 0, 0)_4$ model diagnosed for the variable (GNP) to be forecast.

Output 9.2.1. ARIMAX Specification before Event Input Selection

```
                              The HPFDIAGNOSE Procedure

                              ARIMA Model Specification

         Functional                                             Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

GNP      NONE       YES       0  2  1  2  0  0           4 RMSE        21.2180

                              ARIMA Model Specification

                              Variable Status

                              GNP      OK
```

Output 9.2.2 shows that the SHOCK and SHIFT events are significant.

Output 9.2.2. ARIMA Event Selection

```
                                ARIMA Event Selection

             Event
             Name     Selected      d     D    Status

             SHOCK    YES           2     0    OK
             SHIFT    YES           2     0    OK
```

Output 9.2.3 shows that the input variable, CONSUMP, is selected in the model.

Output 9.2.3. ARIMA Input Selection

```
                               ARIMA Input Selection

 Input                   Functional
 Variable    Selected    Transform     d    D   Delay   Numerator   Denominator

 CONSUMP     YES         NONE          1    0       0           1             1
 INVEST      NO          NONE          1    0       0           1             2
 EXPORTS     NO          NONE          1    0       2           1             0
 GOVT        NO          NONE          1    0       5           2             1

                               ARIMA Input Selection

                            Input
                            Variable    Status

                            CONSUMP     OK
                            INVEST      UnStable
                            EXPORTS     UnStable
                            GOVT        UnStable
```

Output 9.2.4 shows that the RMSE model selection criterion with the events and input is smaller than that without the events and input.

Output 9.2.4. ARIMAX Specification after Event Input Selection

```
         ARIMA Model Specification After Adjusting for Events and Inputs

          Functional                                                   Model
Variable Transform  Constant  p  d  q  P  D  Q Seasonality Input Event Criterion

GNP      NONE       YES       0  2  1  2  0  0            4     1     2 RMSE

                         ARIMA Model Specification After
                         Adjusting for Events and Inputs

                     Variable Statistic    Status

                     GNP        15.3422    OK
```

Example 9.3. Intermittent Demand Series

This example shows that the data is an intermittent demand series.

```
data inventory;
   input tires @@;
datalines;
0 0 0 6 0 4 0 0 0 2 0 2 2 0 0 0 6 0 0 0
;
```

```
proc hpfdiag data=inventory print=all;
   forecast tires;
run;
```

Output 9.3.1 shows that the variable (TIRES) to be forecast is an intermittent demand series. The Interval/Size demand model and Average demand model were diagnosed to the data. The value of model selection criterion (RMSE) of the Average demand model is smaller than that of the Interval/Size demand model.

Output 9.3.1. Intermittent Demand Model Specification

```
                              The HPFDIAGNOSE Procedure

                         Intermittent Demand Model Specification

          Demand     Functional  Selected                Model
Variable  Model      Transform   Model       Component   Criterion   Statistic

tires     INTERVAL   NONE        DAMPTREND   LEVEL       RMSE           0.8754
                                             TREND
                                             DAMP
          SIZE       LOG         LINEAR      LEVEL
                                             TREND
          AVERAGE    NONE        LINEAR      LEVEL                      0.6125
                                             TREND
```

Example 9.4. Exponential Smoothing Model

This example illustrates the use of exponential smoothing models (ESM).

```
data investment;
   input inv @@;
   label inv="Gross Investment";
datalines;
33.1 45. 77.2 44.6 48.1 74.4 113. 91.9 61.3 56.8 93.6
159.9 147.2 146.3 98.3 93.5 135.2 157.3 179.5 189.6
;


proc hpfdiag data=investment print=all;
   forecast inv;
   esm;
run;
```

Output 9.4.1 shows that the variable (INV) to be forecast diagnosed the damped-trend exponential smoothing model.

Output 9.4.1. Exponential Smoothing Model Specification

```
                              The HPFDIAGNOSE Procedure

                      Exponential Smoothing Model Specification

             Functional    Selected                      Model
Variable     Transform     Model          Component      Criterion     Statistic

inv          NONE          DAMPTREND      LEVEL          RMSE            26.2492
                                          TREND
                                          DAMP
```

Example 9.5. Unobserved Components Model

This example illustrates the use of the UCM statement in the HPFDIAGNOSE procedure.

```
data ozone;
   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
 .    .    .    .    .    .    .    .    .    .    .    .
;
```

```
    proc hpfdiag data=ozone print=all;
       id date interval=month;
       forecast ozone;
       input x1 summer winter;
       ucm;
   run;
```

Output 9.5.1 shows that the input SUMMER is selected in the model.

Output 9.5.1. UCM Input Selection

```
                              The HPFDIAGNOSE Procedure

                                 UCM Input Selection

 Input                        Functional
 Variable     Selected        Transform       d      D     Delay     Status

 x1           NO              NONE            0      0        11     Not Improved
 summer       YES             NONE            0      0         6     OK
 winter       NO                                                     Not Significant
```

Output 9.5.2 shows that the variable to be forecast is explained by the irregular, level and season components, and an input.

Output 9.5.2. Unobserved Components Model Specification

```
                    Unobserved Components Model(UCM) Specification

           Functional                                                     Model
 Variable  Transform    Component    Selected  Stochastic  Seasonality   Criterion

 ozone     NONE         IRREGULAR    YES       YES                       RMSE
                        LEVEL        YES       NO
                        SLOPE        NO
                        SEASON       YES       YES                  12
                        INPUT        1

                           Unobserved Components Model(UCM)
                                    Specification

                     Variable  Statistic    Status

                     ozone        0.9912    OK
```

References

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.

Choi, ByoungSeon (1992), *ARMA Model Identification*, New York: Springer-Verlag, 129-132.

Cobb, G.W. (1978), "The Problem of the Nile: Conditional Solution to a Change Point Problem," *Biometrika,* 65, 243-251.

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.

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.

Hasza, D.P. and Fuller, W.A. (1979), "Estimation for Autoregressive Processes with Unit Roots," *The Annals of Statistics,* 7, 1106-1120.

Hasza, D.P., and Fuller, W.A. (1984), "Testing for Nonstationary Parameter Specifications in Seasonal Time Series Models," *The Annals of Statistics,* 10, 1209-1216.

Hillmer, S.C., Larcker, D.F., and Schroeder, D.A. (1983), "Forecasting accounting data: A multiple time-series analysis," *Journal of Forecasting,* 2, 389-404.

de Jong, P. and Penzer, J. (1998), "Diagnosing Shocks in Time Series," *Journal of the American Statistical Association*, Vol. 93, No. 442.

McKenzie, Ed (1984). "General exponential smoothing and the equivalent ARMA process," *Journal of Forecasting*, 3, 333-344.

Tsay, R.S. and Tiao, G.C. (1984), "Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation Function for Stationary and Nonstationary ARMA Models," *Journal of the American Statistical Association,* 79 (385), 84-96.

Chapter 10
The HPFENGINE Procedure

Chapter Contents

Chapter 10
The HPFENGINE Procedure

Overview

The HPFENGINE procedure provides an automatic way to generate forecasts for many time series or transactional data in one step. The procedure can automatically choose the best forecast model from a user-defined model list. Specifications for the candidate forecast models are independent of any data series and can be generated by the user or chosen from a default set. Supported model families include the following:

- Smoothing
- Intermittent Demand
- Unobserved Component
- ARIMA

Additionally, you can provide user-defined forecasts with data drawn from a SAS data set or through the use of user-written functions.

All parameters associated with the forecast model are optimized based on the data. The HPFENGINE procedure selects the appropriate model for each data series based on one of several model selection criteria.

The procedure operates in a variety of modes. At its most comprehensive, all appropriate candidate models from a list are fit to a particular series and the model that produces the best fit, based on a user-determined criterion, is determined. Forecasts are then produced from this model. It is also possible to skip the selection process and fit a particular model and produce subsequent forecasts. Finally, given a set of parameter estimates and model specifications, the procedure will bypass the fit stage entirely and calculate forecasts directly.

The HPFENGINE procedure writes the time series extrapolated by the forecasts, the series summary statistics, the forecasts and confidence limits, the parameter estimates, and the fit statistics to output data sets.

The HPFENGINE procedure can forecast both time series data, whose observations are equally spaced at a specific time interval, or transactional data, whose observations are not spaced at any particular time interval. For transactional data, the data are accumulated at a specified time interval to form a time series.

Getting Started

In the simplest usage, the HPFENGINE procedure produces results similar to those of the HPF procedure:

```
proc hpfengine data=sashelp.air print=(select summary);
   id date interval=month;
   forecast air;
run;
```

The GLOBALSELECTION= and REPOSITORY= options assume their respective default values. Therefore automatic selection is performed for the AIR series in SASHELP.AIR by using the models specified in the BEST selection list. The selection list BEST is found, together with the smoothing models it references, in SASHELP.HPFDFLT.

The results of the automatic model selection are displayed in Figure 10.1.

```
                          The HPFENGINE Procedure

                      Model Selection Criterion = RMSE

   Model      Statistic    Selected     Label

   smsimp         .        Removed      Simple Exponential Smoothing
   smdoub         .        Removed      Double Exponential Smoothing
   smdamp         .        Removed      Damped-Trend Exponential Smoothing
   smlin          .        Removed      Linear Exponential Smoothing
   smadwn     12.245596    No           Winters Method (Additive)
   smwint     10.579085    Yes          Winters Method (Multiplicative)
   smseas     14.169905    No           Seasonal Exponential Smoothing
```

Figure 10.1. Selection Results

The first four models in the selection list are nonseasonal smoothing models. The HPFENGINE procedure determined that the series AIR in SASHELP.AIR was seasonal and attempted to fit only seasonal models.

The multiplicative Winters method produced a fit with the smallest root mean square error (RMSE).

As another example, consider the problem of comparing the performance of multiple ARIMA models for a particular series. This is done by using the HPFARIMASPEC procedure to specify the models and grouping together references to those models in a selection list. Selection lists are built with the HPFSELECT procedure, as shown in the following program. Selection results and the forecast summary are shown in Figure 10.2.

```
proc hpfarimaspec repository=sasuser.mycat
                  name=arima1;
   dependent symbol=air q=1 diflist=1 12 noint transform=log;
run;
```

```
proc hpfarimaspec repository=sasuser.mycat
                  name=arima2;
   dependent symbol=air q=(1,12) diflist=1 12 noint transform=log;
run;

proc hpfselect repository=sasuser.mycat
               name=myselect;
      select select=rmse holdout=0;
   spec arima1 arima2;
run;

proc hpfengine data=sashelp.air print=(select forecasts)
               repository=sasuser.mycat globalselection=myselect;
   id date interval=month;
   forecast air;
run;
```

```
                            The HPFENGINE Procedure

                             Variable Information

    Name                                                                  AIR
    Label                               international airline travel (thousands)
    First                                                             JAN1949
    Last                                                              DEC1960
    Number of Observations Read                                           144


                        Model Selection Criterion = RMSE

Model    Statistic  Selected  Label

ARIMA1   11.891584  No        ARIMA: Log( AIR ) ~ D = (1,12)  Q = 1   NOINT
ARIMA2   11.059916  Yes       ARIMA: Log( AIR ) ~ D = (1,12)  Q = (1,12)   NOINT


                          Forecasts for Variable AIR

                                     Standard
    Obs        DATE      Forecasts      Error       95% Confidence Limits

    145    JAN1961       450.3513     17.3713       417.2608       485.3439
    146    FEB1961       425.6158     20.5676       386.7242       467.3272
    147    MAR1961       478.6340     27.0051       427.8837       533.7048
    148    APR1961       501.0466     31.8162       441.5784       566.2403
    149    MAY1961       512.5109     35.8088       445.9079       586.2004
    150    JUN1961       584.8311     44.2775       502.8483       676.3035
    151    JUL1961       674.9025     54.7586       573.9404       788.4345
    152    AUG1961       667.8935     57.5888       562.1344       787.6937
    153    SEP1961       558.3906     50.8322       465.3910       664.4679
    154    OCT1961       499.4696     47.7516       412.4199       599.4142
    155    NOV1961       430.1668     43.0040       352.0416       520.4285
    156    DEC1961       479.4592     49.9391       389.0353       584.5587
```

Figure 10.2. Selection and Forecast Results

Syntax

The following statements are used with the HPFENGINE procedure:

PROC HPFENGINE *options*;
  BY *variables*;
  ID *variable* **INTERVAL=** *interval options*;
  FORECAST *variable-list / options*;
  INPUT *variable-list / options*;
  STOCHASTIC *variable-list / options*;
  CONTROL *variable-list / options*;
  ADJUST *variable* **= (** *variable-list* **)** */ options*;
  EXTERNAL *variable-list / options* **;**
  SCORE ;

Functional Summary

The statements and options controlling the HPFENGINE procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify variables to forecast	FORECAST	
specify the time ID variable	ID	
specify input variables	INPUT	
specify input variables	STOCHASTIC	
specify input variables	CONTROL	
specify input forecasts, bounds, PIs	EXTERNAL	
request creation of score files	SCORE	
Model Selection Options		
specify location of model repository	PROC HPFENGINE	REPOSITORY=
specify model selection list	PROC HPFENGINE	GLOBALSELECTION=
Data Set Options		
specify the input data set	PROC HPFENGINE	DATA=
specify the mapping/estimate input data set	PROC HPFENGINE	INEST=
specify the events data set	PROC HPFENGINE	EVENT=

Description	Statement	Option
specify the output data set	PROC HPFENGINE	OUT=
specify the mapping/estimate output data set	PROC HPFENGINE	OUTTEST
specify the forecast output data set	PROC HPFENGINE	OUTSTAT
specify the forecast component output data set	PROC HPFENGINE	OUTCOMPONENT
replace missing values	PROC HPFENGINE	REPLACEMISSING
Accumulation Options		
accumulation frequency	ID	INTERVAL=
length of seasonal cycle	PROC HPFENGINE	SEASONALITY=
interval alignment	ID	ALIGN=
starting time ID value	ID	START=
ending time ID value	ID	END=
date format	ID	FORMAT=
accumulation statistic	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	ACCUMULATE=
missing value interpretation	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	SETMISSING=
zero value interpretation	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	ZEROMISS=
trim missing values	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	TRIMMISS=

Description	Statement	Option
Forecasting Horizon Options		
specify data to hold back	PROC HPFENGINE	BACK=
specify forecast horizon or lead	PROC HPFENGINE	LEAD=
Forecasting Control Options		
specify forecasting control options	PROC HPFENGINE	TASK=
Scoring Options		
specify location of score repository	PROC HPFENGINE	SCOREREPOSITORY=
Printing and Plotting Options		
specify graphical output	PROC HPFENGINE	PLOT=
specify printed output	PROC HPFENGINE	PRINT=
specify detailed printed output	PROC HPFENGINE	PRINTDETAILS
Miscellaneous Options		
specify that analysis variables are processed in sorted order	PROC HPFENGINE	SORTNAMES
specify error printing options	PROC HPFENGINE	ERRORCONTROL=

PROC HPFENGINE Statement

PROC HPFENGINE *options* **;**

The following options can be used in the PROC HPFENGINE statement.

BACK= *n*
specifies the number of observations before the end of the data that the multistep forecasts are to begin. This option is often used to obtain performance statistics. See the PRINT= option details about printing performance statistics. The default is BACK=0.

DATA= *SAS-data-set*
names the SAS data set containing the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

GLOBALSELECTION= *catalog name*
specifies the name of a catalog entry that serves as a model selection list. This is the

selection list used to forecast all series if no INEST= data set is given. It is also the selection list used if individual model selections are missing in the INEST= data set when INEST= is provided. If REPOSITORY= is not present, GLOBALSELECTION defaults to BEST, specified in SASHELP.HPFDFLT.

INEST= *SAS-data-set*
contains information that maps forecast variables to models or selection lists, and data set variables to model variables. It may also contain parameter estimates used if the TASK=FORECAST option is present. INEST= is optional. See the description of GLOBALSELECTION= for more information.

INEVENT= *SAS-data-set*
contains information describing predefined events. This data set is usually created by the HPFEVENTS procedure. This option is only used if events are included in a model.

LEAD= *n*
specifies the number of periods ahead to forecast (forecast lead or horizon). The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation 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.

OUT= *SAS-data-set*
names the output data set to contain the forecasts of the variables specified in the subsequent FORECAST statements. If an ID variable is specified, it will also be included in the OUT= data set. The values are accumulated based on the ACCUMULATE= option and forecasts are appended to these values. If the OUT= data set is not specified, an default output data set DATAn is created. If you do not want the OUT= data set created, then use OUT=_NULL_.

OUTCOMPONENT= *SAS-data-set*
names the output data set to contain the forecast components. The components included in the output depend on the model.

OUTEST= *SAS-data-set*
contains information that maps forecast variables to model specifications, and data set variables to model variables and parameter estimates.

An OUTEST= data set will frequently be used as the INEST= data set for subsequent invocations of PROC HPFENGINE. In such a case, if the FORECAST statement option TASK=FORECAST is used, forecasts are generated using the parameter estimates found in this data set and are not reestimated.

OUTFOR= *SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, and prediction standard error). The OUTFOR= data set is particularly useful for displaying the forecasts in tabular or graphical form.

OUTSTAT= *SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is particularly useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series.

PRINT= *option*
specifies the printed output desired. By default, the HPFENGINE procedure produces no printed output.

The following printing options are available:

ESTIMATES	prints the results of parameter estimation.
FORECASTS	prints the forecasts.
STATISTICS	prints the statistics of fit.
SUMMARY	prints the forecast summary.
PERFORMANCE	prints the performance statistics for each forecast.
PERFORMANCESUMMARY	prints the performance summary for each BY group.
PERFORMANCEOVERALL	prints the performance summary for all of the BY groups.
SELECT	prints the label and fit statistics for each model in the selection list.
BIAS	prints model bias information.
COMPONENTS	prints forecast model components.
CANDIDATES	prints parameter estimates for each candidate model fit to the series forecast series.
ALL	Same as specifying PRINT=(ESTIMATES SELECT FORECASTS STATISTICS BIAS). PRINT=(ALL CANDIDATES COMPONENTS PERFORMANCE PERFORMANCESUMMARY PERFORMANCEOVERALL) prints all the options listed.

PRINTDETAILS
specifies that output requested with the PRINT= option be printed in greater detail.

REPOSITORY= *catalog name*
is a two-level SAS catalog name specifying the location of the model repository. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=. The default for this option is SASHELP.HPFDFLT.

SCOREREPOSITORY= *catalog name | libref*
is a two-level SAS catalog name specifying the location of the model score repository. This repository is where score files are written if the SCORE statement is used in the HPFENGINE procedure. There is no default score repository. The presence of a SCORE statement requires that the SCOREREPOSITORY= option also be present.

SEASONALITY= *n*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means

that every group of three observations forms a seasonal cycle. The SEASONALITY= option is applicable only for seasonal forecasting models. By default, the length of the seasonal cycle is 1 (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 12.

SORTNAMES

specifies that the variables specified in the FORECAST statement are processed in sorted order.

TASK= *option*

controls the model selection and parameter estimation process. Available options are as follows:

SELECT — performs model selection, estimates parameters of the selected model, and produces forecasts. This is the default.

SELECT*(options)* — performs model selection, estimates parameters of the selected model, produces forecasts, and potentially overrides settings in the model selection list. If a selection list does not specify a particular item, and that item is specified with a TASK=SELECT option, the value as set in TASK=SELECT is used. If an option is specified in selection list, the corresponding value set in TASK=SELECT is not used unless the OVERRIDE option is also present. The available options for TASK=SELECT are as follows:

HOLDOUT= *n* — specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series ending at the last nonmissing observation.

HOLDOUTPCT= *number* — specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$, where T is the length of the time series with beginning and ending missing values removed.

CRITERION= *option* — specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. The following list shows the valid values for the CRITERION= option and the statistics of fit these option values specify:

SSE	Sum of Square Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
UMSE	Unbiased Mean Square Error
URMSE	Unbiased Root Mean Square Error
MAXPE	Maximum Percent Error
MINPE	Minimum Percent Error

MPE	Mean Percent Error
MAPE	Mean Absolute Percent Error
MDAPE	Median Absolute Percent Error
GMAPE	Geometric Mean Absolute Percent Error
MINPPE	Minimum Predictive Percent Error
MAXPPE	Maximum Predictive Percent Error
MSPPE	Mean Predictive Percent Error
MAPPE	Symmetric Mean Absolute Predictive Percent Error
MDAPPE	Median Absolute Predictive Percent Error
GMAPPE	Geometric Mean Absolute Predictive Percent Error
MINSPE	Minimum Symmetric Percent Error
MAXSPE	Maximum Symmetric Percent Error
MSPE	Mean Symmetric Percent Error
SMAPE	Symmetric Mean Absolute Percent Error
MDASPE	Median Absolute Symmetric Percent Error
GMASPE	Geometric Mean Absolute Symmetric Percent Error
MINRE	Minimum Relative Error
MAXRE	Maximum Relative Error
MRE	Mean Relative Error
MRAE	Mean Relative Absolute Error
MDRAE	Median Relative Absolute Error
GMRAE	Geometric Mean Relative Absolute Error
MAXERR	Maximum Error
MINERR	Minimum Error
ME	Mean Error
MAE	Mean Absolute Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion

SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion

INTERMITTENT= *number* specifies a number greater than one that is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number, then the series is assumed to be intermittent.

SEASONTEST= *option* specifies the options related to the seasonality test.

The following values for the SEASONTEST= option are allowed:

NONE	No test.
(SIGLEVEL=number)	Significance probability value to use in testing whether seasonality is present in the time series. The value must be between 0 and 1.

A smaller value of the SIGLEVEL= option means that stronger evidence of a seasonal pattern in the data is required before PROC HPFENGINE will use seasonal models to forecast the time series. The default is SEASONTEST=(SIGLEVEL=0.01).

ALPHA= *number* specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1. As an example, ALPHA=0.05 produces 95% confidence intervals.

OVERRIDE forces the use of any options listed.

NOALTLIST By default, if none of the models in a selection list can be successfully fit to a series, PROC HPFENGINE returns to the selection list SASHELP.HPFDFLT.BEST and restarts the selection process. The NOALTLIST option disables this action and sets the forecast to missing if no models can be fit from the initial selection list. There will be an observation in OUTSUM=, if requested, corresponding to the variable and BY group in question, and the _STATUS_ variable will be nonzero.

MINOBS=(TREND= *n*) Normally the models in a selection list are not subset by trend. Using the MINOBS=(TREND=) option, a user can specify that no trend model be fit to any series with fewer than *n* nonmissing values.

Incorporation of a trend is checked only for smoothing, UCM, and ARIMA models. For the smoothing case, only simple smoothing is a non-trend model. For UCM, the absence of a slope component qualifies it as a non-trend model. For ARIMA, there must be no differencing of the dependent variable for PROC HPFENGINE to consider it a non-trend model.

The value of n must be greater than or equal to 1. The default is n = 1.

MINOBS=(SEASON= *n*) specifies that no seasonal model be fit to any series with fewer observations than n multiplied by the seasonal cycle length. The value of n must be greater than or equal to 1. The default is n = 2.

MINOBS=*n* The MINOBS= option enables the user to specify that any series with fewer than *n* nonmissing values not be fit using the models in the selection list, but instead be forecast as the mean of the observations in the series. The value of n must be greater than or equal to 1. The default is n = 1.

FIT estimates parameters by using the model specified in the INEST= data set, then forecast. No model selection is performed.

FIT*(options)* estimates parameters by using the model specified in the INEST= data set, then forecast, potentially overriding the significance level in the model selection list used to compute forecast confidence intervals. No model selection is performed. If a selection list does not specify alpha, and alpha is specified in the TASK=FIT option, the value as set in TASK=FIT will be in effect. If alpha is specified in selection list, the corresponding value set in TASK=FIT will not be used unless the OVERRIDE option is also present. The available options for TASK=FIT are as follows:

ALPHA= *number* specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1.

OVERRIDE forces the use of any options listed.

UPDATE estimates parameters by using the model specified in the INEST= data set, then forecast. TASK=UPDATE differs from TASK=FIT in that the parameters found in the INEST= data set are used as starting values in the estimation. No model selection is performed.

UPDATE*(options)* estimates parameters by using the model specified in the INEST= data set, using the parameter estimates as starting values, then forecast, potentially overriding the significance level in the model selection list used to compute forecast confidence intervals.

No model selection is performed. If a selection list does not specify alpha, and alpha is specified in the TASK=UPDATE option, the value as set in TASK=UPDATE will be in effect. If alpha is specified in the selection list, the corresponding value set in TASK=UPDATE will not be used unless the OVERRIDE option is also present. The available options for TASK=UPDATE are as follows:

ALPHA= *number*	specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1.
OVERRIDE	forces the use of any options listed.

FORECAST — forecasts using model and parameters specified in the INEST= data set. No parameter estimation occurs.

FORECAST*(options)* — forecasts using model and parameters specified in the INEST= data set, potentially overriding the significance level in the model selection list used to compute forecast confidence intervals. No parameter estimation occurs. If a selection list does not specify alpha, and alpha is specified in the TASK=FORECAST option, the value as set in TASK=FORECAST will be used. If alpha is specified in the selection list, the corresponding value set in TASK=FORECAST will not be used unless the OVERRIDE option is also present. The available options for TASK=FORECAST are as follows:

ALPHA= *number*	specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1.
OVERRIDE	forces the use of any options listed.

ERRORCONTROL= (SEVERITY= (*severity-options* **) STAGE= (** *stage-options* **) MAXMESSAGE=** *number* **)**

allows finer control of message printing. The error severity level and HPFENGINE procedure processing stages are set independently. A logical 'and' is taken over all the specified options, and any message that tests true against the results of the 'and' is printed.

Available *severity-options* are as follows:

LOW	specifies low severity, minor issues
MEDIUM	specifies medium severity problems
HIGH	specifies severe errors
ALL	specifies all severity levels of LOW, MEDIUM, and HIGH

NONE	specifies that no messages from the HPFENGINE procedure are printed

Available *stage-options* are as follows:

PROCEDURELEVEL	specifies that the procedure stage is option processing and validation
DATAPREP	specifies the accumulation of data and the application of SETMISS= and ZEROMISS= options
SELECTION	specifies the model selection process
ESTIMATION	specifies the model parameter estimation process
FORECASTING	specifies the model evaluation and forecasting process
ALL	specifies all PROCEDURELEVEL, DATAPREP, SELECTION, ESTIMATION, and FORECASTING options

Examples are as follows:

```
errorcontrol=(severity=(high medium) stage=all);
```

prints high- and moderate-severity errors at any processing stage of PROC HPFENGINE.

```
errorcontrol=(severity=high stage=dataprep);
```

prints high-severity errors only during the data preparation.

```
errorcontrol=(severity=none stage=all);
```

turns off messages from PROC HPFENGINE.

```
errorcontrol=( severity=(high medium low)
               stage=(procedurelevel dataprep selection estimation forecasting)
```

specifies the default behavior. Also the following code specifies the default behavior:

```
errorcontrol=(severity=all stage=all)
```

BY Statement

BY *variables;*

A BY statement can be used with PROC HPFENGINE to obtain separate analyses for groups of observations defined by the BY variables.

FORECAST Statement

FORECAST *variable-list / options*;

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast.

A data set variable can be specified in only one FORECAST statement. Any number of FORECAST statements can be used. The following options can be used with the FORECAST statement.

ACCUMULATE= *option*
specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

REPLACEMISSING
specifies that embedded missing actual values are replaced with one-step-ahead forecasts in the OUT= data set.

SETMISSING= *option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS= *option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. If the TRIMMISS= option is not specified in the FORECAST statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS= *option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

ID Statement

ID *variable 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 actual time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the actual time series. The information specified affects all variables specified in subsequent FORECAST 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 actual time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations coinciding with a particular time period (e.g., 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 nonmissing 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 option and the START= option can be used to ensure that data associated with each BY group contain the same number of observations.

FORMAT= *option*
specifies a SAS format used for the DATE variable in the output data sets. The default format is the same as that of the DATE variable in the DATA= data set.

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. 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. See *SAS/ETS User's Guide* for the intervals that can be specified.

SETMISSING= *option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series. If a number is specified, missing values are set to that 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 the absence of 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 nonmissing value.
LAST	Missing values are set to the accumulated last nonmissing value.
PREVIOUS | PREV	Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

If SETMISSING=MISSING is specified and the MODEL= option specifies a smoothing model, the missing observations are smoothed over. If MODEL=IDM is specified, missing values are assumed to be periods of no demand; that is, SETMISSING=MISSING is equivalent to SETMISSING=0.

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 END= value, the series is truncated. This option and the END= option can be used to ensure that data associated with each BY group contain the same number of observations.

TRIMMISS= *option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the FORECAST statement. The following options are provided:

NONE	No missing value trimming is applied.
LEFT	Beginning missing values are trimmed.
RIGHT	Ending missing values are trimmed.
BOTH	Both beginning and ending missing values are trimmed. This is the default.

ZEROMISS= *option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following options can also be used to determine how beginning and/or ending zero values are assigned:

NONE	Beginning and/or ending zeros are unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing and/or zero, the series is not changed.

INPUT Statement

INPUT *variable-list / options*

The INPUT statement lists the numeric variables in the DATA= data set whose accumulated values will be used as deterministic input in the forecasting process.

Future values for input variables must be supplied. If future values are unknown, consider using either the STOCHASTIC statement or the CONTROL statement. If it will be necessary to later modify future values using the forecast score function HPFSCSUB, use the CONTROL statement.

A data set variable can be specified in only one INPUT statement. Any number of INPUT statements can be used.

The following options can be used with the INPUT statement:

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the INPUT statement. If the ACCUMULATE= option is not specified in the INPUT statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

REQUIRED=*YES | NO*

enables or disables a check of inputs to models. The kinds of problems checked include the following:

- errors in functional transformation
- an input consisting of only a constant value or all missing values
- errors introduced by differencing
- multicollinearity among inputs

If REQUIRED=YES, these checks are not performed and no inputs are dropped from a model. The model might subsequently fail to fit during parameter estimation or forecasting for any of the reasons in the preceding list.

If REQUIRED=NO, inputs are checked and those with errors, or those judged collinear, are dropped from the model for the current series and task only. No changes are persisted in the model specification.

This option has no effect on models with no inputs.

The default is REQUIRED=YES.

SETMISSING=*option | number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for the variables listed in the INPUT statement. If the SETMISSING= option is not specified in the INPUT statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS=*option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the INPUT statement.

If the TRIMMISS= option is not specified in the INPUT statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS=*option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the INPUT statement. If the ZEROMISS= option is not specified in the INPUT statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

STOCHASTIC Statement

STOCHASTIC *variable-list / options*;

The STOCHASTIC statement lists the numeric variables in the DATA= data set whose accumulated values will be used as stochastic input in the forecasting process.

Future values of stochastic inputs need not be provided. By default, they are automatically forecast using one of the following smoothing models.

- Simple
- Double
- Linear
- Damped trend
- Seasonal
- Winters method (additive and multiplicative)

The model with the smallest in-sample MAPE is used to forecast the future values of the stochastic input.

A data set variable can be specified in only one STOCHASTIC statement. Any number of STOCHASTIC statements can be used.

The following options can be used with the STOCHASTIC statement:

ACCUMULATE=*option*
specifies how the data set observations are accumulated within each time period for the variables listed in the STOCHASTIC statement. If the ACCUMULATE= option is not specified in the STOCHASTIC statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

SELECTION=*option*
specifies the selection list used to forecast the stochastic variables. The default is BEST, found in SASHELP.HPFDFLT.

REQUIRED=*YES | NO*
enables or disables a check of inputs to models. The kinds of problems checked include the following:

- errors in functional transformation
- an input consisting of only a constant value or all missing values
- errors introduced by differencing
- multicollinearity among inputs

If REQUIRED=YES, these checks are not performed and no inputs are dropped from a model. The model might subsequently fail to fit during parameter estimation or forecasting for any of the reasons in the preceding list.

If REQUIRED=NO, inputs are checked and those with errors, or those judged collinear, are dropped from the model for the current series and task only. No changes are persisted in the model specification.

This option has no effect on models with no inputs.

The default is REQUIRED=YES.

REPLACEMISSING
specifies that embedded missing actual values be replaced with one-step-ahead forecasts in the STOCHASTIC variables.

SETMISSING=*option | number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the STOCHASTIC statement. If the SETMISSING= option is not specified in the STOCHASTIC statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS=*option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the STOCHASTIC statement.

If the TRIMMISS= option is not specified in the STOCHASTIC statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS=*option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the STOCHASTIC statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

CONTROL Statement

CONTROL *variable-list / options***;**

The CONTROL statement lists the numeric variables in the DATA= data set whose accumulated values will be used as input in the forecasting process. The future values of the control variables in the forecast statement are determined by the EXTEND= option. Only input values used in a CONTROL statement are adjustable in the score evaluation subroutine HPFSCSUB.

The following options can be used with the CONTROL statement:

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the CONTROL statement. If the ACCUMULATE= option is not specified in the CONTROL statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

EXTEND=*option*

specifies how future values of the control variables are set. The following options are provided:

NONE	Future values are set to missing.
AVERAGE	Future values are set to the mean of the values in the fit range. This is the default.
FIRST	Future values are set to the first value found in the fit range.
LAST	Future values are set to the last value found in the fit range.
MINIMUM	Future values are set to the minimum of the values in the fit range.
MAXIMUM	Future values are set to the maximum of the values in the fit range.
MEDIAN	Future values are set to the median of the values in the fit range.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the CONTROL statement. If the SETMISSING= option is not specified in the CONTROL statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS=*option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the CONTROL statement.

If the TRIMMISS= option is not specified in the CONTROL statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS=*option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the CONTROL statement. If the ZEROMISS= option is not specified in the CONTROL statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

ADJUST Statement

ADJUST *variable = (variable-list) / options***;**

The ADJUST statement lists the numeric variables in the DATA= data set whose accumulated values will be used to adjust the dependent values. Adjustments can be performed before and/or after forecasting. A particular forecast variable can be referenced by multiple forecast statements.

The first numeric variable listed is the variable to which adjustments specified in that statement will apply. This variable must appear in a FORECAST statement. The numeric variables used as the source of the adjustments are listed following the parentheses. Options determine which adjustments will be applied and when they will be applied. More information about the use of adjustments is in the details section.

The following options can be used with the ADJUST statement:

OPERATION=(*preadjust***,** *postadjust***)**

specifies how the adjustments are applied to the forecast variable. The *preadjust* option determines how the adjustment variables are applied to the dependent variable prior to forecasting. The *postadjust* option determines how the adjustment variables are applied to the forecast results.

Computations with missing values are handled differently in the adjustment statement than in other parts of SAS. If any of the adjustment operations result in a nonmissing dependent value being added to, subtracted from, divided by, or multiplied by a missing value, the nonmissing dependent value is left unchanged. Division by zero produces a missing value.

The following predefined adjustment operations are provided:

NONE	No adjustment operation is performed. This is the default.
ADD	Variables listed in the adjustment statement are added to the dependent variable.
SUBTRACT	Variables listed in the adjustment statement are subtracted from the dependent variable.

MULTIPLY	Dependent variable is multiplied by variables listed in the adjustment statement.
DIVIDE	Dependent variable is divided by variables listed in the adjustment statement.
MIN	Dependent variable is set to the minimum of the dependent variable and all variables listed in the adjustment statement.
MAX	Dependent variable is set to the maximum of the dependent variable and all variables listed in the adjustment statement.

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the ADJUST statement. If the ACCUMULATE= option is not specified in the CONTROL statement, accumulation is determined by the ACCUMULATE= option of the ID statement. See the ID statement ACCUMULATE= option for more details.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the ADJUST statement. If the SETMISSING= option is not specified in the ADJUST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS=*option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the ADJUST statement.

If the TRIMMISS= option is not specified in the ADJUST statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS=*option*

specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the ADJUST statement. If the ZEROMISS= option is not specified in the CONTROL statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

EXTERNAL Statement

EXTERNAL *variable-list / options***;**

The EXTERNAL statement lists the numeric variables in the DATA= data set whose accumulated values are used as predicted values for an external model, and possibly prediction standard errors and lower and upper confidence intervals.

Any variables used in an EXMMAP in a selection list, as specified by PROC HPFSELECT, must appear in an EXTERNAL statement.

The following options can be used with the EXTERNAL statement:

SETMISSING=*option* | *number*
specifies how missing values (either actual or accumulated) are assigned in the accumulated time series for variables listed in the EXTERNAL statement. If the SETMISSING= option is not specified in the EXTERNAL statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRIMMISS=*option*
specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series for variables listed in the EXTERNAL statement.

If the TRIMMISS= option is not specified in the EXTERNAL statement, missing values are set based on the TRIMMISS= option of the ID statement. See the ID statement TRIMMISS= option for more details.

ZEROMISS=*option*
specifies how beginning and/or ending zero values (either actual or accumulated) are interpreted in the accumulated time series for variables listed in the ADJUST statement. If the ZEROMISS= option is not specified in the EXTERNAL statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

SCORE Statement

The SCORE statement, used in conjunction with one or more FORECAST statements, causes the generation of score files. The score files are written to the location specified by SCOREREPOSITORY=.

Details

The HPFENGINE procedure can be used to forecast time series data as well as transactional data. If the data are transactional, then the procedure must first accumulate the data into a time series before the data can be forecast. The procedure uses the following sequential steps to produce forecasts:

1. Accumulation
2. Missing Value Interpretation
3. Pre-forecast Adjustment
4. Diagnostic Tests
5. Model Selection
6. Transformations
7. Parameter Estimation
8. Forecasting

9. Inverse Transformation
10. Post-forecast Adjustment
11. Statistics of Fit

These steps are described in the following sections.

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 actual 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 preceding observations fall within the 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:

```
O1MAR1999    14.14
O1APR1999     .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contained no missing values, the accumulated time series might have missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPFENGINE procedure can effectively handle missing values. But sometimes missing values are known, such as when missing values are created from accumulation, and no observations should be interpreted as no (zero) value. 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.

Adjustment Operations

Pre-adjustment variables can be used to adjust the dependent series prior to model parameter estimation, evaluation, and forecasting. After the predictions of the adjusted dependent series are obtained from the forecasting mode, the post-adjustment variables can be used to adjust these forecasts to obtain predictions that more closely match the original dependent series.

Pre-adjustment (Before Forecasting) Step

If y_t is the dependent series and $x_{i,t}$ for $i = 1, \ldots, M$ are the M adjustment series, the adjusted dependent series, w_t, is as follows:

$$\begin{aligned} w_{i,t} &= op_i^b(y_t, x_{i,t-k_i}) \text{ for } i = 1 \\ w_{i,t} &= op_i^b(w_{i-1,t}, x_{i,t-k_i}) \text{ for } 1 < i \leq M \\ w_t &= w_{i,t} \text{ for } i = M \end{aligned}$$

where op_i^b represents the pre-adjustment operator and k_i is the time shift for the adjustment series $x_{i,t}$.

As can be seen, the pre-adjustment operators are nested and applied sequentially from $i = 1, \ldots, M$.

Pre-adjustment is performed on the historical data only.

Adjusted Forecast Step

$$\begin{aligned} w_t &= \hat{F}(w_t) + \varepsilon_t \text{ historical data} \\ \hat{w}_t &= \hat{F}(w_t) \text{ historical data and forecast horizon} \end{aligned}$$

where $\hat{F}()$ represents the fitted forecasting function.

Post-adjustment (After Forecasting) Step

$$\begin{aligned} \hat{w}_{i,t} &= op_i^a(\hat{w}_t, x_{i,t-k_i}) \text{ for } i = M \\ \hat{w}_{i,t} &= op_i^a(\hat{w}_{i+1,t}, x_{i,t-k_i}) \text{ for } 1 \leq i < M \\ \hat{y}_t &= \hat{w}_{i,t} \end{aligned}$$

where op_i^a represents the post-adjustment operator for the adjustment series $x_{i,t}$.

As can be seen, the post-adjustment operators are nested and applied sequentially from $i = M, \ldots, 1$, which is the reverse of the pre-adjustment step.

Post-adjustment is performed on the historical data as well as the forecast horizon.

Notes

Typically the pre-adjustment, op_i^b, operator and post-adjustment, op_i^a, operators are inverses of each other, that is, $op_i^a = inverse(op_i^b)$.

For example, if the pre-adjustment operator is subtraction, the post-adjustment operator is addition, as shown in the following:

$$\begin{aligned} w_t &= y_t - \sum_{i=1}^{M} x_{i,t} \\ \hat{y}_t &= \hat{w}_t + \sum_{i=1}^{M} x_{i,t} \end{aligned}$$

For example, if the pre-adjustment operator is division, the post-adjustment operator is multiplication, as shown in the following:

$$w_t = y_t \prod_{i=1}^{M} \frac{1}{x_{i,t}}$$

$$\hat{y}_t = \hat{w}_t \prod_{i=1}^{M} x_{i,t}$$

Pre-adjustment is often followed by post-adjustment, but the inverse operation is not required. It is acceptable to pre-adjust, but not post-adjust, and vice versa.

As an example, the following statement adds, before forecasting, the values contained in the variables firstadj and scndadj to the dependent variable air. After forecasting air, there is no post-adjustment. The variable air must be specified in a FORECAST statement.

```
ADJUST air=(firstadj scndadj) / OPERATION = (ADD,NONE);
```

Diagnostic Tests

Diagnostic test control is specified in the model selection list and can be set by using the HPFSELECT procedure. The INTERMITTENT= option in the HPFSELECT procedure's DIAGNOSE statement sets the threshold for categorizing a series as intermittent or nonintermittent. Likewise, the SEASONTEST= option in the HPFSELECT procedure's DIAGNOSE statement sets the threshold for categorizing a series as seasonal or nonseasonal.

Model Selection

When more than one candidate model is specified in a model selection list, forecasts for each candidate model are compared using the model selection criterion specified by the CRITERION= option in the SELECT statement in PROC HPFSELECT.

The selection criterion is computed using the multistep forecasts in the holdout sample range if the HOLDOUT= or HOLDOUTPCT= options are specified, or the one-step-ahead forecasts for the full range of the time series if the HOLDOUT= and HOLDOUTPCT= options are not specified. The candidate model with the best selection criterion is selected to forecast the time series.

Transformations

If a forecasting model specifies a transformation of the dependent series, the time series is transformed prior to model parameter estimation and forecasting. Only strictly positive series can be transformed.

Parameter Estimation

All parameters associated with the model are optimized based on the data with the default parameter restrictions imposed. If a forecasting model specifies a transformation of the dependent series, the transformed time series data are used to estimate the model parameters.

Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. For the intermittent demand models, specified missing values are assumed to be periods of no demand. For other models, 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. See the "Forecasting" section for more information about how missing values are treated in the smoothing models.

The treatment of missing values can also be specified by the user with the SETMISSING= option, which changes the missing values prior to modeling.

Even though all of the observed data are nonmissing, using the ACCUMULATE= option can create missing values in the accumulated series.

Forecasting

After the model parameters are estimated, one-step-ahead forecasts are generated for the full range of the actual (optionally transformed) time series data, and multistep forecasts are generated from the end of the observed time series to the future time period after the last observation specified by the LEAD= option. If there are missing values at the end of the time series, the forecast horizon will be greater than that specified by the LEAD= option.

Inverse Transformations

If a forecasting model specifies a transformation of the dependent series, the forecasts of the transformed time series are inverse transformed. By default, the mean (expected) forecasts are generated. If the MEDIAN option is specified in one of the model specification procedures, the median forecasts are generated.

Statistics of Fit

The statistics of fit (or goodness-of-fit statistics) are computed by comparing the actual time series data and the generated forecasts. If the dependent series was transformed according to the model specification, the statistics of fit are based on the inverse transformed forecasts.

Data Set Input/Output

The HPFENGINE procedure can create the OUT=, OUTEST=, OUTFOR=, OUTCOMPONENT=, and OUTSTAT= data sets. In general, if the forecasting process for a particular time series fails, the output corresponding to this series is not recorded or is set to missing in the relevant output data set, and appropriate error and/or warning messages are recorded in the log.

INEST= Data Set

The INEST= data set contains the variables specified in the BY statement as well as the following variables:

NAME	Variable name
SELECT	Name of selection list
MODEL	Name of model
MODELVAR	Model variable mapping
DSVAR	Data set variable mapping

The referenced selection lists, taken together with the data set to model variable mappings, drive the forecasting process.

If the FORECAST statement TASK = FORECAST option is specified, other variables must be present in INEST=. The additional variables are as follows:

TRANSFORM	Transformation applied
COMPONENT	Model component (e.g., AR, MA, Trend, etc.)
FACTOR	Model factor
LAG	Lag of input
SHIFT	Shift
PARM	Parameter name

LABEL	Parameter label
EST	Parameter estimate
STDERR	Parameter estimate standard error
TVALUE	Parameter estimate *t*-value
PVALUE	Parameter estimate *p*-value
STATUS	Indicates success/failure in estimating parameter

OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, and FORECAST statements. If the ID statement is specified, the ID variables are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the FORECAST statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. If the REPLACEMISSING option is specified, embedded missing values are replaced by the one-step-ahead forecasts. If any of the forecasting steps fail for a particular variable, the variable values are extended by missing values.

OUTFOR= Data Set

The OUTFOR= data set contains the variables specified in the BY statement as well as the following variables:

NAME	Variable name
TIMEID	Time ID values
PREDICT	Predicted values
STD	Prediction standard errors
LOWER	Lower confidence limits
UPPER	Upper confidence limits
ERROR	Prediction errors

If the forecasting step fails for a particular variable, no observations are recorded. If the TRANSFORM= option is specified, the values in the preceding variables are the inverse transformed forecasts. If the MEDIAN option is specified, the median forecasts are stored; otherwise, the mean forecasts are stored.

OUTEST= Data Set

The OUTEST= data set contains the variables specified in the BY statement as well as the following variables:

NAME	Variable name
SELECT	Name of selection list
MODEL	Name of model

MODELVAR	Model variable mapping
DSVAR	Data set variable mapping
TRANSFORM	Transformation applied
COMPONENT	Model component (e.g., AR, MA, Trend, etc.)
FACTOR	Model factor
LAG	Lag of input
SHIFT	Shift
PARM	Parameter name
LABEL	Parameter label
EST	Parameter estimate
STDERR	Parameter estimate standard error
TVALUE	Parameter estimate *t*-value
PVALUE	Parameter estimate *p*-value
STATUS	Indicates success/failure in estimating parameter

An OUTEST= data set is frequently used as the INEST= data set for subsequent invocations of PROC HPFENGINE. In such a case, if the option TASK=FORECAST is used, forecasts are generated using the parameter estimates found in this data set as opposed to being reestimated.

OUTCOMPONENT= Data Set

The OUTCOMPONENT= data set contains the variables specified in the BY statement as well as the following variables:

NAME	Variable name
COMP	Name of the component
TIME	Time ID
ACTUAL	Dependent series value
PREDICT	Component forecast
LOWER	Lower confidence limit
UPPER	Upper confidence limit
STD	Prediction standard error

OUTSTAT= Data Set

The OUTSTAT= data set contains the variables specified in the BY statement as well as the following variables. The following variables contain observations related to the statistics of fit step:

NAME	Variable name
SSE	Sum of squares error
MSE	Mean square error
UMSE	Unbiased mean square error
RMSE	Root mean square error
URMSE	Unbiased root mean square error
MAPE	Mean absolute percent error
MAE	Mean absolute error
RSQUARE	R-Square
ADJRSQ	Adjusted R-square
AADJRSQ	Amemiya's adjusted R-square
RWRSQ	Random walk R-square
AIC	Akaike information criterion
SBC	Schwarz Bayesian information criterion
APC	Amemiya's prediction criterion
MAXERR	Maximum error
MINERR	Minimum error
MINPE	Minimum percent error
MAXPE	Maximum percent error
ME	Mean error
MPE	Mean Percent error

If the statistics of fit step fails for a particular variable, no observations are recorded.

ODS Table Names

Table 1 relates the PRINT= options to ODS tables.

Table 10.1. ODS Tables Produced in PROC HPFENGINE

ODS Table Name	Description	Specific Models
ODS Tables Created by the PRINT=SUMMARY Option		
Variable	Forecast Variable Information	
ForecastSummary	Forecast Summary	
ODS Tables Created by the PRINT=ESTIMATES Option		
Variable	Forecast Variable Information	
ParameterEstimates	Parameter Estimates	
ODS Tables Created by the PRINT=SELECT Option		
Variable	Forecast Variable Information	
ModelSelection	Model Selection Statistics	

Table 10.1. (continued)

ODS Table Name	Description	Specific Models
ODS Tables Created by the PRINT=FORECASTS Option		
Variable	Forecast Variable Information	
Forecasts	Forecast	
Demands	Demands	IDM models only
DemandSummary	Demand Summary	IDM models only
ODS Tables Created by the PRINT=STATISTICS Option		
Variable	Forecast Variable Information	
FitStatistics	Statistics of Fit	
ODS Tables Created by the PRINT=BIAS Option		
Variable	Forecast Variable Information	
TestUnbiasedness	Bias Test	
ParameterEstimates	Bias Test Parameter Estimates	
ODS Tables Created by the PRINT=CANDIDATES Option		
Variable	Forecast Variable Information	
ParameterEstimates	Parameter Estimates	
ODS Tables Created by the PRINT=COMPONENTS Option		
Variable	Forecast Variable Information	
ComponentEstimates	Parameter Estimates	
ODS Tables Created by the PRINT=PERFORMANCE Option		
Variable	Forecast Variable Information	
Performance	Performance Statistics	
ODS Tables Created by the PRINT=PERFORMANCESUMMARY Option		
Variable	Forecast Variable Information	
PerformanceSummary	Performance Summary	
ODS Tables Created by the PRINT=PERFORMANCEOVERALL Option		
Variable	Forecast Variable Information	
PerformanceSummary	Performance Overall	

The ODS table ForecastSummary is related to all time series within a BY group. The other 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 HPFENGINE procedure. These graphics are experimental, 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= option in the PROC HPFENGINE statement according to the following syntax.

PLOT= *option* **|** **(***options***)**

specifies the graphical output desired. By default, the HPFENGINE procedure produces no graphical output. The following printing options are available:

ERRORS	plots prediction error time series graphics.
ACF	plots prediction error autocorrelation function graphics.
PACF	plots prediction error partial autocorrelation function graphics.
IACF	plots prediction error inverse autocorrelation function graphics.
WN	plots white noise graphics.
FORECASTS	plots forecast graphics.
FORECASTSONLY	plots the forecast in the forecast horizon only.
COMPONENTS	plots the forecast components.
CANDIDATES	plots model and error graphics for each candidate model fit to the series forecast series.
ALL	specifies all of the preceding PLOT= options.

For example, PLOT=FORECASTS plots the forecasts for each series. The PLOT= option produces printed output for these results by using the Output Delivery System (ODS). The PLOT= statement is experimental for this release of SAS.

ODS Graph Names

PROC HPFENGINE assigns a name to each graph it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 10.2.

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you specify the PLOT= option in the PROC HPFENGINE statement.

Table 10.2. ODS Graphics Produced by PROC HPFENGINE

ODS Graph Name	Plot Description	Statement	PLOT= Option
CandidateErrorHoldoutPlot	Plot of Candidate Model Errors with Holdout	PROC HPFENGINE	PLOT=CANDIDATES
CandidateErrorPlot	Plot of Candidate Model Errors	PROC HPFENGINE	PLOT=CANDIDATES
CandidateModelHoldoutPlot	Plot of Candidate Models with Holdout	PROC HPFENGINE	PLOT=CANDIDATES
CandidateModelPlot	Plot of Candidate Models	PROC HPFENGINE	PLOT=CANDIDATES
ComponentEstimatesPlot	Plot of Component Estimates	PROC HPFENGINE	PLOT=COMPONENTS
DemandErrorsPlot	Average Demand Errors	PROC HPFENGINE	PLOT=ERRORS
DemandForecastsPlot	Average Demand Forecasts	PROC HPFENGINE	PLOT=FORECASTS
DemandIntervalHistogram	Demand Interval Histogram	PROC HPFENGINE	PLOT=ALL
DemandIntervalPlot	Demand Interval Forecast Plot	PROC HPFENGINE	PLOT=ALL

Table 10.2. (continued)

ODS Graph Name	Plot Description	Statement	Option
DemandSizeHistogram	Demand Size Histogram	PROC HPFENGINE	PLOT=MODELS
DemandSizePlot	Demand Size Forecast Plot	PROC HPFENGINE	PLOT=MODELS
ErrorACFNORMPlot	Standardized autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=ACF
ErrorACFPlot	Autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=ACF
ErrorHistogram	Prediction Error Histogram	PROC HPFENGINE	PLOT=ERRORS
ErrorIACFNORMPlot	Standardized inverse autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=IACF
ErrorIACFPlot	Inverse autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=IACF
ErrorPACFNORMPlot	Standardized partial autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=PACF
ErrorPACFPlot	Partial autocorrelation of Prediction Errors	PROC HPFENGINE	PLOT=PACF
ErrorPlot	Plot of Prediction Errors	PROC HPFENGINE	PLOT=ERRORS
ErrorWhiteNoiseLogProbPlot	White noise log probability plot of Prediction Errors	PROC HPFENGINE	PLOT=WN
ErrorWhiteNoisePlot	White noise plot of Prediction Errors	PROC HPFENGINE	PLOT=ALL
ErrorWhiteNoiseProbPlot	White noise probability plot of Prediction Errors	PROC HPFENGINE	PLOT=WN
ForecastsOnlyPlot	Forecasts Only Plot	PROC HPFENGINE	PLOT=FORECASTSONLY
ForecastsPlot	Forecasts Plot	PROC HPFENGINE	PLOT=FORECAST
ModelForecastsPlot	Model and Forecasts Plot	PROC HPFENGINE	PLOT=ALL
ModelPlot	Model Plot	PROC HPFENGINE	PLOT=ALL
StockingAveragePlot	Stocking Average Plot	PROC HPFENGINE	PLOT=FORECASTS
StockingLevelPlot	Stocking Level Plot	PROC HPFENGINE	PLOT=FORECASTS

Examples

Example 10.1. The TASK Option

The default selection list is used in this example. The first call to the HPFENGINE procedure uses the default TASK = SELECT action. A model is selected from the default list, parameters are estimated, and a forecast is produced.

The second call to PROC HPFENGINE uses the model selected in the first call, estimates parameters, and produces forecasts.

The final call to PROC HPFENGINE reuses the model parameters estimates to forecast the series.

This example demonstrates that consistent forecast results are produced in each of the three PROC HPFENGINE runs. Many applications only call PROC HPFENGINE once, with TASK = SELECT, to produce forecasts.

Selection results and forecast summary are shown in Output 10.1.1.

```
proc hpfengine data=sashelp.air outfor=outselect outest=outest
               print=(select forecasts);
   id date interval=month;
   forecast air;
run;

proc hpfengine data=sashelp.air inest=outest outfor=outfit;
   id date interval=month;
   forecast air / task = fit;
run;

proc hpfengine data=sashelp.air inest=outest outfor=outforecast;
   id date interval=month;
   forecast air / task = forecast;
run;

proc compare base=outselect compare=outfit briefsummary;
run;

proc compare base=outselect compare=outforecast briefsummary;
run;
```

Output 10.1.1. Selection and Forecast Results

```
                              The HPFENGINE Procedure

                         Model Selection Criterion = RMSE

     Model      Statistic    Selected    Label

     smsimp       .          Removed     Simple Exponential Smoothing
     smdoub       .          Removed     Double Exponential Smoothing
     smdamp       .          Removed     Damped-Trend Exponential Smoothing
     smlin        .          Removed     Linear Exponential Smoothing
     smadwn     12.245596    No          Winters Method (Additive)
     smwint     10.579085    Yes         Winters Method (Multiplicative)
     smseas     14.169905    No          Seasonal Exponential Smoothing



                               The COMPARE Procedure
                  Comparison of WORK.OUTSELECT with WORK.OUTFIT
                                   (Method=EXACT)

NOTE: No unequal values were found. All values compared are exactly equal.



                               The COMPARE Procedure
                Comparison of WORK.OUTSELECT with WORK.OUTFORECAST
                                   (Method=EXACT)

NOTE: No unequal values were found. All values compared are exactly equal.
```

Example 10.2. Different Types of Input

This example demonstrates the use of different input types in the HPFENGINE procedure. The output is shown in Output 10.2.1.

```
data seriesj;
   input x y @@;
   label x = 'Input Gas Rate'
         y = 'Output CO2';
   date = intnx( 'day', '01jan1950'd, _n_-1 );
   format date DATE.;

/* list only part of data */
datalines;
-0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5
 0.373  53.4  0.441  53.1  0.461  52.7  0.348  52.4
 0.127  52.2 -0.180  52.0 -0.588  52.0 -1.055  52.4
-1.421  53.0 -1.520  54.0 -1.302  54.9 -0.814  56.0
   :
   :
;
```

```
* make spec;
proc hpfarimaspec repository=sasuser.mycat
                  name=arimasp;
   dependent symbol=Y p=2;
   input     symbol=X num=2 den=1 lag=3;
run;

* make selection list;
proc hpfselect repository=sasuser.mycat
               name=myselect;
   spec arimasp;
run;

data seriesj_trunc;
   set seriesj;
   if (date >= '17oct1950'd) then y = .;
run;

* future values of input are given;
proc hpfengine data=seriesj_trunc outfor=outfor
               lead=7
               repository=sasuser.mycat
               globalselection=myselect;
   id       date interval=day;
   forecast y;
   input    x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) noobs;
   title2 'Results with deterministic input';
   var date predict upper lower;
run;

data seriesj_trunc;
   set seriesj;
   if (date < '17oct1950'd);
run;

* future values of input are automatically forecast;
proc hpfengine data=seriesj_trunc outfor=outfor
               lead=7
               repository=sasuser.mycat
               globalselection=myselect;
   id         date interval=day;
   forecast   y;
   stochastic x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) noobs;
   title2 'Results with stochastic input';
   var date predict upper lower;
run;

* future values of input are taken as the average of past values;
proc hpfengine data=seriesj_trunc outfor=outfor
               lead=7
               repository=sasuser.mycat
               globalselection=myselect;
   id       date interval=day;
   forecast y;
   control  x / extend=average;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) noobs;
   title2 'Results with controllable input';
   var date predict upper lower;
run;
```

Output 10.2.1. Selection and Forecast Results

```
                         Results with stochastic input

                        date      PREDICT      UPPER       LOWER

                   17OCT1950     57.0684     57.5116     56.6252
                   18OCT1950     56.4050     57.1794     55.6307
                   19OCT1950     55.3432     56.3382     54.3481
                   20OCT1950     54.1800     55.2879     53.0721
                   21OCT1950     53.1714     54.3183     52.0246
                   22OCT1950     52.4126     53.5648     51.2603
                   23OCT1950     51.9055     53.0582     50.7529




                        Results with controllable input

                        date      PREDICT      UPPER       LOWER

                   17OCT1950     57.0684     57.5116     56.6252
                   18OCT1950     56.4050     57.1794     55.6307
                   19OCT1950     55.3432     56.3382     54.3481
                   20OCT1950     54.3326     55.4405     53.2247
                   21OCT1950     53.5819     54.7288     52.4351
                   22OCT1950     53.2104     54.3627     52.0582
                   23OCT1950     53.0632     54.2159     51.9105
```

Example 10.3. Incorporating Events

This example creates an event called PROMOTION. The event is added as a simple regressor to each ARIMA specification in the selection list. The output is shown in Output 10.3.1.

```
* make data set;
data work_intv;
   set sashelp.workers;
   if date >= '01oct80'd then electric = electric+100;
   drop masonry;
run;

* define event 'promotion';
proc hpfevents data=work_intv lead=12;
   id date interval=month;
   eventdef promotion=  '01oct80'd / TYPE=LS;
   eventdata out= evdsout1 (label='list of events');
   eventdummy out= evdumout1 (label='list of events');
run;

* make specs;
proc hpfarimaspec repository=sasuser.mycat
              name=sp1
              label="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
   dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
run;

proc hpfarimaspec repository=sasuser.mycat
              name=sp2
              label="ARIMA(2,1,0)(1,1,0)_12 No Intercept";
   dependent symbol=Y p=(1, 2)(12) diflist=1 12 noint;
run;

* make selection list;
proc hpfselect repository=sasuser.mycat
               name=myselect
               label="My Selection List";
   select select=mape holdout=12;
   spec sp1 sp2 /
        inputmap(symbol=Y data=electric)
        eventmap(symbol=_NONE_ event=promotion);
run;

* select, fit and forecast;
proc hpfengine data=work_intv
     globalselection=myselect repository=sasuser.mycat
     print=(select estimates) inevent=evdsout1;
   id date interval=month;
   forecast electric;
run;
```

Output 10.3.1. Selection and Forecast Results

```
                              The HPFENGINE Procedure

                               Variable Information

           Name                                                ELECTRIC
           Label                               electrical workers, thousands
           First                                                JAN1977
           Last                                                 JUL1982
           Number of Observations Read                               67


                          Model Selection Criterion = MAPE

      Model      Statistic     Selected     Label

      SP1        3.3275773     No           ARIMA(0,1,2)(0,1,1)_12 No Intercept
      SP2        1.8745987     Yes          ARIMA(2,1,0)(1,1,0)_12 No Intercept


                               Parameter Estimates

                                                     Standard                  Approx
    Component     Parameter       Estimate           Error     t Value     Pr > |t|

    ELECTRIC      AR1_1            0.38213         0.14776        2.59       0.0127
    ELECTRIC      AR1_2            0.01383         0.14711        0.09       0.9255
    ELECTRIC      AR2_12          -0.53640         0.14980       -3.58       0.0008
    PROMOTION     SCALE           95.98147         2.97398       32.27       <.0001
```

Example 10.4. Using the SCORE Statement

This example demonstrates the use of the SCORE statement to create a forecast score file. The output is shown in Output 10.4.1.

```
data air;
   set sashelp.air;
   controlinput = log(air);
run;

proc hpfarimaspec repository=work.repo name=ar;
   dependent symbol=Y q=1 dif=12;
   input predefined=LINEAR;
   input symbol=controlinput;
   input predefined=INVERSE;
run;

proc hpfselect repository=work.repo name=select;
   spec ar;
run;

* generate score;
proc hpfengine data=air repository=work.repo out=engineout
      globalselection=select scorerepository=work.scores;
   id date interval=month;
   forecast air;
   controllable controlinput / extend=avg;
   score;
run;

filename score catalog "work.scores.scor0.xml";

proc means data=air mean noprint;
   var controlinput;
   output out=controlmean mean=mean;
run;

data _null_;
   set controlmean;
   call symput("mean", mean);
run;

data forecasts;
   drop p1 p2 p3;
   format date monyy.;
   date = '01jan1961'd;
   call HPFSCSUB('score',3,'CONTROLINPUT',&mean,&mean,&mean,
                 'PREDICT',p1,p2,p3);
   forecast = p1; date = intnx('month', date, 0); output;
   forecast = p2; date = intnx('month', date, 1); output;
   forecast = p3; date = intnx('month', date, 1); output;
run;

data compare;
   merge engineout forecasts;
   by date;
run;

proc print data=compare(where=(forecast ne .)) noobs;
run;
```

Output 10.4.1. Selection and Forecast Results

```
                                DATE       AIR       forecast

                             JAN1961    416.408    416.408
                             FEB1961    391.715    391.715
                             MAR1961    419.312    419.312
```

Example 10.5. HPFENGINE and HPFDIAGNOSE Procedures

The HPFDIAGNOSE procedure is often used in conjunction with the HPFENGINE procedure. This example demonstrates the interaction between the two. The output is shown in Output 10.5.1.

```
proc hpfdiagnose data=sashelp.air
                 repository=sasuser.mycat
                 outest=est;
   id date interval=month;
```

```
      forecast air;
      arimax;
      ucm;
      esm;
   run;

   proc hpfengine data=sashelp.air inest=est outest=outest
                  repository=sasuser.mycat
                  print=(select estimates summary) lead=4;
      id date interval=month;
      forecast air;
   run;
```

Output 10.5.1. Selection and Forecast Results

```
                             The HPFENGINE Procedure

                              Variable Information

    Name                                                                   AIR
    Label                              international airline travel (thousands)
    First                                                              JAN1949
    Last                                                               DEC1960
    Number of Observations Read                                            144


                         Model Selection Criterion = RMSE

                         Model      Statistic    Selected

                         diag68     10.835333    No
                         diag69     10.652082    Yes
                         diag70     10.980119    No

                        Model Selection Criterion = RMSE

       Model     Label

       diag68    ARIMA: Log( AIR ) ~ P = 1  D = (1,12)  Q = (12)   NOINT
       diag69    Log Winters Method (Multiplicative)
       diag70    UCM: Log( AIR ) = TREND + SEASON + ERROR


                               Parameter Estimates

                                                    Standard                Approx
Component    Parameter            Estimate         Error     t Value    Pr > |t|

AIR          Level Weight          0.40151       0.03602       11.15      <.0001
AIR          Trend Weight        0.0010000     0.0074742        0.13      0.8938
AIR          Seasonal Weight       0.63549       0.07525        8.44      <.0001


                                Forecast Summary

   Variable     Value                JAN1961     FEB1961     MAR1961     APR1961

   AIR          Predicted           451.3518    427.3583    490.3388    512.3521
```

Example 10.6. The ADJUST Statement

This example illustrates the use of a pre-adjustment addition operation. The output is shown in Output 10.6.1.

```
data items;
   set sashelp.air;
   adjair = air;
   adjvar = mod(_N_-1, 12) + 1;
run;

* forecast both air and adjair;
proc hpfengine data=items outfor=outfor;
   id date interval=month;
   forecast air adjair;
   adjust  adjair = (adjvar ) / operation=(ADD, NONE);
run;

proc print data=outfor(where=(actual=.));
   title 'Dependent variable adjust in ADJUST statement';
   var _NAME_ DATE PREDICT;
run;

* adjust adjair outside of HPFENGINE;
data items;
   set items;
   adjair = air + adjvar;
run;

proc hpfengine data=items outfor=outfor;
   id date interval=month;
   forecast adjair;
run;

proc print data=outfor(where=(actual=.));
   title 'Dependent variable adjust in data step';
   var _NAME_ DATE PREDICT;
run;
```

Output 10.6.1. Selection and Forecast Results

```
                  Dependent variable adjust in ADJUST statement

                       Obs    _NAME_       DATE     PREDICT

                       145    AIR        JAN1961    445.297
                       146    AIR        FEB1961    418.143
                       147    AIR        MAR1961    464.089
                       148    AIR        APR1961    494.026
                       149    AIR        MAY1961    504.958
                       150    AIR        JUN1961    572.595
                       151    AIR        JUL1961    662.704
                       152    AIR        AUG1961    653.774
                       153    AIR        SEP1961    545.893
                       154    AIR        OCT1961    487.715
                       155    AIR        NOV1961    415.259
                       156    AIR        DEC1961    459.607
                       301    adjair     JAN1961    445.790
                       302    adjair     FEB1961    419.644
                       303    adjair     MAR1961    466.855
                       304    adjair     APR1961    497.088
                       305    adjair     MAY1961    509.258
                       306    adjair     JUN1961    577.941
                       307    adjair     JUL1961    668.646
                       308    adjair     AUG1961    660.870
                       309    adjair     SEP1961    554.217
                       310    adjair     OCT1961    497.321
                       311    adjair     NOV1961    426.593
                       312    adjair     DEC1961    472.292



                      Dependent variable adjust in data step

                       Obs    _NAME_       DATE     PREDICT

                       145    adjair     JAN1961    445.790
                       146    adjair     FEB1961    419.644
                       147    adjair     MAR1961    466.855
                       148    adjair     APR1961    497.088
                       149    adjair     MAY1961    509.258
                       150    adjair     JUN1961    577.941
                       151    adjair     JUL1961    668.646
                       152    adjair     AUG1961    660.870
                       153    adjair     SEP1961    554.217
                       154    adjair     OCT1961    497.321
                       155    adjair     NOV1961    426.593
                       156    adjair     DEC1961    472.292
```

Chapter 11
The HPFESMSPEC Procedure

Chapter Contents

Chapter 11

The HPFESMSPEC Procedure

Overview

The HPFESMSPEC procedure creates model specifications files for exponential smoothing models (ESM).

You can specify many types of exponential models using this procedure. In particular, any model that can be analyzed using the HPF procedure can be specified.

Getting Started

The following example shows how to create an exponential smoothing model specification file. In this example, a model specification for a Winters method is created.

```
proc hpfesmspec repository=sasuser.mymodels
                name=mywinters
                label="Winters Method";
  esm method=winters
  ;
run;
```

The options in the PROC HPESMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog SASUSER.MYMODELS, the NAME= option specifies that the name of the file be "mywinters.xml", and the LABEL= option specifies a label for this catalog member. The ESM statement in the procedure specifies the exponential smoothing model and the options used to control the parameter estimation process for the model.

Syntax

The following statements are used with the HPFESMSPEC procedure.

PROC HPFESMSPEC *options*;
  ESM *options*;

Functional Summary

The statements and options controlling the HPFESMSPEC procedure are summarized in the following table.

Description	Statement	Option
Statements		
specifies the exponential smoothing model	ESM	
Model Repository Options		
specifies the model repository	PROC HPFESMSPEC	REPOSITORY=
specifies the model specification name	PROC HPFESMSPEC	NAME=
specifies the model specification label	PROC HPFESMSPEC	LABEL=
Exponential Smoothing Model Options		
specifies the time series transformation	ESM	TRANSFORM=
specifies median forecasts	ESM	MEDIAN
specifies the time series forecasting model	ESM	METHOD=
specifies that the smoothing model parameters are fixed values	ESM	NOEST
specifies that stable parameter estimates are not required	ESM	NOSTABLE
specifies the model selection criterion	ESM	SELECT=
specifies the level weight parameter initial value	ESM	LEVELPARM=
specifies the level weight parameter restrictions	ESM	LEVELREST=
specifies the trend weight parameter initial value	ESM	TRENDPARM=
specifies the trend weight parameter restrictions	ESM	TRENDREST=
specifies the damping weight parameter initial value	ESM	DAMPPARM=
specifies the damping weight parameter restrictions	ESM	DAMPREST=
specifies the season weight parameter initial value	ESM	SEASONPARM=
specifies the season weight parameter restrictions	ESM	SEASONREST=

PROC HPFESMSPEC Statement

PROC HPFESMSPEC *options ;*

The following options can be used in the PROC HPFESMSPEC statement.

LABEL= *SAS-label*
specifies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

NAME= *SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

REPOSITORY= *SAS-catalog-name|SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

ESM Statement

ESM *options ;*

The ESM statement is used to specify an exponential model.

The following examples illustrate typical uses of the ESM statement:

```
/* default specification */
esm;

/* Simple Exponential Smoothing */
esm method=simple;

/* Double Exponential Smoothing */
esm method=double;

/* Linear Exponential Smoothing */
esm method=linear;

/* Damp-Trend Exponential Smoothing */
esm method=damptrend;

/* Seasonal Exponential Smoothing */
esm method=seasonal;

/* Winters Method */
esm method=winters;

/* Additive-Winters Method */
esm method=addwinters;

/* Best Smoothing Model */
esm method=best;

/* Best Non-Seasonal Smoothing Model */
esm method=bestn;

/* Best Seasonal Smoothing Model */
```

```
esm method=bests;

/* Log Simple Exponential Smoothing */
esm method=simple transform=log;

/* Log Double Exponential Smoothing */
esm method=double transform=log;

/* Log Linear Exponential Smoothing */
esm method=linear transform=log;

/* Log Damp-Trend Exponential Smoothing */
esm method=damptrend transform=log;

/* Log Seasonal Exponential Smoothing */
esm method=seasonal transform=log;

/* Log Winters Method */
esm method=winters transform=log;

/* Log Additive-Winters Method */
esm method=addwinters transform=log;

/* Best Log Smoothing Model */
esm method=best transform=log;

/* Best Log Non-Seasonal Smoothing Model */
esm method=bestn transform=log;

/* Best Log Seasonal Smoothing Model */
esm method=bests transform=log;
```

The following example illustrates how to automatically choose the exponential smoothing model using MAPE as the model selection criterion:

```
esm method=simple transform=auto select=mape
```

The preceding example fits two forecast models (simple and log simple exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

```
esm method=seasonal transform=auto select=mape
```

The preceding example fits two forecast models (seasonal and log seasonal exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

```
esm method=best transform=auto select=mape
```

The preceding example fits 14 forecast models (best and log best exponential smoothing) to the time series. The forecast model that results in the lowest MAPE is used to forecast the time series.

The default specification selects the best exponential smoothing model without transformation (METHOD=BEST TRANSFORM=NONE).

The following options can be specified in the ESM statement:

TRANSFORM= *option*

specifies the time series transformation to be applied to the 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
AUTO	Automatically choose between NONE and LOG based on model selection criteria.

When the TRANSFORM= option is specified, the time series must be strictly positive. Once the time series is transformed, the model parameters are estimated using the transformed time series. The forecasts of the transformed time series are then computed, and finally, the transformed time series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

MEDIAN

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series using the TRANSFORM= option, the mean and median time series forecast values are identical.

METHOD= *method-name*

specifies the forecasting model to be used to forecast the time series. A single model can be specified or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the SELECT= option of the ESM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

SIMPLE	Simple (Single) Exponential Smoothing
DOUBLE	Double (Brown) Exponential Smoothing
LINEAR	Linear (Holt) Exponential Smoothing
DAMPTREND	Damped Trend Exponential Smoothing

SEASONAL	Seasonal Exponential Smoothing
WINTERS	Winters Multiplicative Method
ADDWINTERS	Winters Additive Method
BEST	Best Candidate Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND), (SEASONAL, ADDWINTERS, WINTERS)
BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)
BESTS	Best Candidate Smoothing Model (SEASONAL, ADDWINTERS, WINTERS)

NOSTABLE
specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

LEVELPARM= *number*
specifies the level weight parameter initial value. See the following smoothing model parameter specifications options.

LEVELREST=(*number*,*number***)**
specifies the level weight parameter restrictions. See the following smoothing model parameter specifications options.

TRENDPARM= *number*
specifies the trend weight parameter initial value. See the following smoothing model parameter specifications options.

TRENDREST=(*number*,*number***)**
specifies the trend weight parameter restrictions. See the following smoothing model parameter specifications options.

DAMPPARM= *number*
specifies the damping weight parameter initial value. See the following smoothing model parameter specifications options.

DAMPREST=(*number*,*number***)**
specifies the damping weight parameter restrictions. See the following smoothing model parameter specifications options.

SEASONPARM= *number*
specifies the season weight parameter initial value. See the following smoothing model parameter specifications options.

SEASONREST=(*number*,*number***)**
specifies the season weight parameter restrictions. See the following smoothing model parameter specifications options.

NOEST
specifies that the smoothing model parameters are fixed values. To use this option,

all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

SELECT= *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option would often be used in conjunction with the HOLDOUT= option. The default is SELECT=RMSE.

Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified the default is (0.0001 0.9999), which implies that the parameters are restricted between 0.0001 and 0.9999. Parameters and their restrictions are required to be greater than or equal to -1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

Chapter 12
The HPFEVENTS Procedure

Chapter Contents

Chapter 12
The HPFEVENTS Procedure

Overview

The HPFEVENTS procedure provides a way to create and manage events associated with time series for the purpose of analysis. The procedure can create events, read events from an events data set, write events to an events data set, and create dummies based on those events if date information is provided.

A SAS event is used to model any incident that disrupts the normal flow of the process that generated the time series. Examples of commonly used events include natural disasters, retail promotions, strikes, advertising campaigns, policy changes, and data recording error.

An event has a reference name, a date or dates associated with the event, and a set of qualifiers. The event exists separate from any time series; however, the event may be applied to one or more time series. When the event is applied to a time series, a dummy variable is generated that may be used to analyze the impact of the event on the time series. You can use the HPFEVENTS procedure to apply an event or events to a time series, create a dummy variable(s), and save dummy variable(s) in a data set. However, it is not necessary to do this if PROC HPFENGINE or PROC HPFDIAGNOSE will be used to evaluate the time series and the event(s). PROC HPFENGINE and PROC HPFDIAGNOSE create and store the dummy variable(s) in memory for you based on the definition created with PROC HPFEVENTS. You only need to supply the event definition data set created with the EVENTDATA OUT= statement to PROC HPFENGINE or PROC HPFDIAGNOSE.

The advantages of using PROC HPFEVENTS are:

- Dummies generated by PROC HPFEVENTS are automatically extended, shortened, or changed as observations are added and deleted from a time series. Thus, a single EVENT definition can be used for several time series or for different spans of the same series.
- PROC HPFEVENTS can be used to define dummies that function equally well for time series of various intervals, for instance weekly or monthly data. The same EVENT definition can model daily data or weekly totals.
- EVENT definitions can be stored in a data set. EVENT definitions can later be changed, new EVENTS added, or additional dummies generated from an existing data set.
- EVENT definitions stored in a data set can be passed directly to PROC HPFENGINE and PROC HPFDIAGNOSE. Refer to Chapter 10, “The HPFENGINE Procedure,” and Chapter 9, “The HPFDIAGNOSE Procedure,” for details.

- SAS predefined EVENT definitions can be accessed directly from PROC HPFENGINE and PROC HPFDIAGNOSE. See the EVENTKEYstatement for a list of SAS predefined EVENT definitions. Example 12.4 illustrates this feature.
- PROC HPFEVENTS can generate a data set that can be used in other procedures such as PROC REG. Refer to Chapter 66, "The REG Procedure," (*SAS/STAT User's Guide*) . As data is added or deleted from the time series, PROC HPFEVENTS can automatically generate new dummy variables as required.
- PROC HPFEVENTS recognizes predefined variables and dates. Thus, events involving holidays such as Easter and Thanksgiving can be modeled easily, even though the dates of the events change from year to year.

Getting Started

The HPFEVENTS procedure is simple to use. It provides results in output data sets that may be interpreted in other SAS procedures.

The following example will create events and dummies and output the event definitions to a data set named EVDSOUT1 and dummies to a data set named EVDUMOUT1. More examples are shown the "Examples" section on page 438.

```
proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month end='31Dec1952'D;
   eventdef laborday= LABOR / VALUE=2 ;
   eventdef summer= '01Jun1900'D to '01Jun2005'D by year /
                    AFTER=(DURATION=2) LABEL='jun jul aug';
   eventdef yr1950= '01Jan1950'D / PULSE=YEAR ;
   eventdef levelshift= '01Jan1950'D / TYPE=LS ;
   eventdef novdec= CHRISTMAS / BEFORE=(DURATION=1)
                                PULSE=MONTH;
   eventdef first10obs= 1 to 10;
   eventdef everyotherobs= 1 to 200 by 2;
   eventdef saturday= '01Jan1950'D to '31Jan1950'D by WEEK.7;
   eventkey ao15obs;
   eventkey ls01Jan1950D / AFTER=(DURATION=5) ;
   eventkey garbage ;
   eventdata  out= evdsout1 (label='list of events');
   eventdummy  out= evdumout1 (label='dummy variables');
run;
```

Features of PROC HPFEVENTS illustrated by the above statements:

LABORDAY	PROC HPFEVENTS recognizes that "LABOR" is a date keyword. When PROC HPFEVENTS creates a dummy variable for this event, a timing value is generated for each Labor Day that falls in the span of the time series. Each observation that matches the date of Labor Day has a value of 2.

SUMMER	The do-list, '01Jun1900'D to '01Jun2005'D by year, will generate 106 timing values on the first of June for each year from 1900 to 2005. The pulse for each timing value will last for 3 observations, the observation matching June 1st and the two following. For monthly data, this should generate a pulse for June, July, and August of each year from 1900 to 2005. If you add PULSE=MONTH to this statement, then the EVENT always specifies June, July, and August, regardless of the interval of the data. If you specify only the timing value '01Jun1900'D and add PERIOD=YEAR, then you have the same effect for all years, even years before 1900 and after 2005.
YR1950	By specifying a timing value within the year 1950 and using PULSE=YEAR, this event is a pulse for any observations within the year 1950.
LEVELSHIFT	TYPE=LS has, by default, AFTER=(DURATION=ALL). The pulse begins at the observation matching January 1, 1950, and continues to the end of the series. A special missing value of "A" is shown in the _DUR_AFTER_ variable of the events definition data set to represent AFTER=(DURATION=ALL).
NOVDEC	Compare this to the SUMMER event. Here the date keyword CHRISTMAS is used. CHRISTMAS produces the same result as a timing value of '25Decyyyy'D and PERIOD=YEAR. BEFORE=(DURATION=1) and PULSE=MONTH specifies the months of November and December for any year in the span of the series. If the intent is to specify certain months of the year, this is preferable to the syntax used in SUMMER.
FIRST10OBS	Integers in the timing list always specify observation numbers. This dummy is always a pulse from observation 1 to observation 10, regardless of the value of the timing ID variable.
EVERYOTHEROBS	Like FIRST10OBS, this dummy always specifies every other observation starting at observation 1 and ending at observation 199.
SATURDAY	WEEK.7 in the do-list specifies Saturday dates. The do-list produces timing values that are the Saturdays in January of 1950. If the data is daily, there are 4 pulses in January of 1950, one on each Saturday. If the data is weekly, a pulse is formed for 4 successive observations in January of 1950. If the data is monthly and RULE=ADD, which is the default, then the observation for January 1950 counts the number of Saturdays in January.
AO15OBS	AO15OBS is recognized as an event keyword. AO15OBS is a predefined event that means a pulse placed on the 15th observation.
LS01JAN1950D	LS01JAN1950D is recognized as an event keyword. LS01JAN1950D is a predefined event that means a level shift beginning at '01Jan1950'D. The qualifier AFTER=(DURATION=5) modifies the predefined event.

GARBAGE	EVENTKEY GARBAGE is ignored as garbage is not an event keyword. A warning is printed to the log.

```
proc print data=evdsout1;
run;
```

```
proc print data=evdumout1;
run;
```

```
                                                           _
                                             _             D
                                             S             A              _
                          _                  T          _  T        _     D
                          K                  A          E  E        S     T
                    _     E                  R          N  I        T  _  I
   _                C     Y                  T          D  N        A  E  N
   N                L     N                  D          D  T        R  N  T
   A                A     A                  A          A  R        T  D  R
 O M                S     M                  T          T  V        D  D  V
 b E                S     E                  E          E  L        T  T  L
 s _                _     _                  _          _  _        _  _  _

 1 laborday       SIMPLE  LABOR              .          .  .        .  .  .
 2 summer         SIMPLE  .          01JUN1900  01JUN2005  YEAR.6   .  .  .
 3 yr1950         SIMPLE  .          01JAN1950          .  .        .  .  .
 4 levelshift     SIMPLE  .          01JAN1950          .  .        .  .  .
 5 novdec         SIMPLE  CHRISTMAS          .          .  .        .  .  .
 6 first10obs     SIMPLE  .                  .          .  .        .  .  .
 7 everyotherobs  SIMPLE  .                  .          .  .        .  .  .
 8 saturday       SIMPLE  .          07JAN1950  28JAN1950  WEEK1.7  .  .  .
 9 ao15obs        SIMPLE  .                  .          .  .        .  .  .
10 ls01Jan1950D   SIMPLE  .          01JAN1950          .  .        .  .  .


                          _
          _               D _   _      _
    _     O               U D   S      S
    S     B               R U   L      L
    T  _  S               _ R   O      O       _      _
    A  E  I       _ _     B _   P      P    _  T      P _
    R  N  N _     V P     E A   E      E    S  C   _  E L
    T  D  T T     A U     F F   _      _    H  P   R  R A
    O  O  R Y     L L     O T   B      A    I  A   U  I B
 O  B  B  V P     U S     R E   E      F    F  R   L  O E
 b  S  S  L E     E E     E R   F      T    T  M   E  D L
 s  _  _  _ _     _ _     _ _   _      _    _  _   _  _ _

 1  .   . . POINT 2 .     0 0 GROWTH GROWTH 0 0.5 ADD . .
 2  .   . . POINT 1 .     0 2 GROWTH GROWTH 0 0.5 ADD . jun jul aug
 3  .   . . POINT 1 YEAR  0 0 GROWTH GROWTH 0 0.5 ADD . .
 4  .   . . LS    1 .     0 A GROWTH GROWTH 0 0.5 ADD . .
 5  .   . . POINT 1 MONTH 1 0 GROWTH GROWTH 0 0.5 ADD . .
 6  1  10 1 POINT 1 .     0 0 GROWTH GROWTH 0 0.5 ADD . .
 7  1 199 2 POINT 1 .     0 0 GROWTH GROWTH 0 0.5 ADD . .
 8  .   . . POINT 1 .     0 0 GROWTH GROWTH 0 0.5 ADD . .
 9 15   . . POINT 1 .     0 0 GROWTH GROWTH 0 0.5 ADD . .
10  .   . . LS    1 .     0 5 GROWTH GROWTH 0 0.5 ADD . .
```

Figure 12.1. Event Definition Data Set Showing All Variables Related to Event Definitions

```
                                                      e
                                                      v              1
                                                      e              s
                                       l         f    r              0
                                       e         i    y              1
                        l              v         r    o    s         J
                        a              e         s    t    a    a    a
                        b    s    y    l    n    t    h    t    o    n
                        o    u    r    s    o    1    e    u    1    1
            D           r    m    1    h    v    0    r    r    5    9
 O          A     A     d    m    9    i    d    o    o    d    o    5
 b          T     I     a    e    5    f    e    b    b    a    b    0
 s          E     R     y    r    0    t    c    s    s    y    s    D

 1    JAN1949    112    0    0    0    0    0    1    1    0    0    0
 2    FEB1949    118    0    0    0    0    0    1    0    0    0    0
 3    MAR1949    132    0    0    0    0    0    1    1    0    0    0
 4    APR1949    129    0    0    0    0    0    1    0    0    0    0
 5    MAY1949    121    0    0    0    0    0    1    1    0    0    0
 6    JUN1949    135    0    1    0    0    0    1    0    0    0    0
 7    JUL1949    148    0    1    0    0    0    1    1    0    0    0
 8    AUG1949    148    0    1    0    0    0    1    0    0    0    0
 9    SEP1949    136    2    0    0    0    0    1    1    0    0    0
10    OCT1949    119    0    0    0    0    0    1    0    0    0    0
11    NOV1949    104    0    0    0    0    1    0    1    0    0    0
12    DEC1949    118    0    0    0    0    1    0    0    0    0    0
13    JAN1950    115    0    0    1    1    0    0    1    4    0    1
14    FEB1950    126    0    0    1    1    0    0    0    0    0    1
15    MAR1950    141    0    0    1    1    0    0    1    0    1    1
16    APR1950    135    0    0    1    1    0    0    0    0    0    1
17    MAY1950    125    0    0    1    1    0    0    1    0    0    1
18    JUN1950    149    0    1    1    1    0    0    0    0    0    1
19    JUL1950    170    0    1    1    1    0    0    1    0    0    0
20    AUG1950    170    0    1    1    1    0    0    0    0    0    0
21    SEP1950    158    2    0    1    1    0    0    1    0    0    0
22    OCT1950    133    0    0    1    1    0    0    0    0    0    0
23    NOV1950    114    0    0    1    1    1    0    1    0    0    0
24    DEC1950    140    0    0    1    1    1    0    0    0    0    0
25    JAN1951    145    0    0    0    1    0    0    1    0    0    0
26    FEB1951    150    0    0    0    1    0    0    0    0    0    0
27    MAR1951    178    0    0    0    1    0    0    1    0    0    0
28    APR1951    163    0    0    0    1    0    0    0    0    0    0
29    MAY1951    172    0    0    0    1    0    0    1    0    0    0
30    JUN1951    178    0    1    0    1    0    0    0    0    0    0
31    JUL1951    199    0    1    0    1    0    0    1    0    0    0
32    AUG1951    199    0    1    0    1    0    0    0    0    0    0
33    SEP1951    184    2    0    0    1    0    0    1    0    0    0
34    OCT1951    162    0    0    0    1    0    0    0    0    0    0
35    NOV1951    146    0    0    0    1    1    0    1    0    0    0
36    DEC1951    166    0    0    0    1    1    0    0    0    0    0
37    JAN1952    171    0    0    0    1    0    0    1    0    0    0
38    FEB1952    180    0    0    0    1    0    0    0    0    0    0
39    MAR1952    193    0    0    0    1    0    0    1    0    0    0
40    APR1952    181    0    0    0    1    0    0    0    0    0    0
41    MAY1952    183    0    0    0    1    0    0    1    0    0    0
42    JUN1952    218    0    1    0    1    0    0    0    0    0    0
43    JUL1952    230    0    1    0    1    0    0    1    0    0    0
44    AUG1952    242    0    1    0    1    0    0    0    0    0    0
45    SEP1952    209    2    0    0    1    0    0    1    0    0    0
46    OCT1952    191    0    0    0    1    0    0    0    0    0    0
47    NOV1952    172    0    0    0    1    1    0    1    0    0    0
48    DEC1952    194    0    0    0    1    1    0    0    0    0    0
```

Figure 12.2. Event Dummy Data Set

Often a set of EVENT definitions has only a few variables that apply. For instance, in the above example, no datetime timing values are specified, so the variables _STARTDT_, _ENDDT_, and _DTINTRVL_ have all missing values. In such a case, the user can specify the CONDENSE option in the EVENTDATA statement, and PROC HPFEVENTS automatically determines if any variables in the EVENT definition data set contain only the default values; those variables are not included in the output data set. In addition to the missing datetime values, in the example below, the variables _SLOPE_BEF_, _SLOPE_AFT_, _TCPARM_, _RULE_, and _PERIOD_ also contain default values and are omitted from the condensed data set. When PROC HPFEVENTS reads a data set, any variables not in the data set are automatically set to the default value. Thus, it is not necessary to specify CONDENSE when using the EVENTDATA IN= option. PROC HPFEVENTS automatically reads condensed data sets. For more details on the CONDENSE option, see "EVENTDATA OUT= Data Set".

```
proc hpfevents data=sashelp.air ;
   id date interval=month end='31Dec1952'D;
   eventdata  in= evdsout1;
   eventdata  out= evdsout2 condense;
run;


proc print data=evdsout2;
run;
```

```
Obs _NAME_         _KEYNAME_  _STARTDATE_  _ENDDATE_  _DATEINTRVL_  _STARTOBS_

  1 laborday       LABOR                .           .    .                .
  2 summer         .           01JUN1900   01JUN2005    YEAR.6            .
  3 yr1950         .           01JAN1950           .    .                 .
  4 levelshift     .           01JAN1950           .    .                 .
  5 novdec         CHRISTMAS            .           .    .                .
  6 first10obs     .                    .           .    .                1
  7 everyotherobs  .                    .           .    .                1
  8 saturday       .           07JAN1950   28JAN1950    WEEK1.7           .
  9 ao15obs        .                    .           .    .               15
 10 ls01Jan1950D   .           01JAN1950           .    .                 .

                                                 _DUR_   _DUR_
Obs _ENDOBS_ _OBSINTRVL_ _TYPE_ _VALUE_ _PULSE_ BEFORE_ AFTER_ _LABEL_

  1      .         .      POINT      2    .          0       0   .
  2      .         .      POINT      1    .          0       2   jun jul aug
  3      .         .      POINT      1    YEAR       0       0   .
  4      .         .      LS         1    .          0       A   .
  5      .         .      POINT      1    MONTH      1       0   .
  6     10         1      POINT      1    .          0       0   .
  7    199         2      POINT      1    .          0       0   .
  8      .         .      POINT      1    .          0       0   .
  9      .         .      POINT      1    .          0       0   .
 10      .         .      LS         1    .          0       5   .
```

Figure 12.3. Read Previous Event Definition Data Set and Save in Condensed Format

Syntax

The following statements are used with the HPFEVENTS procedure.

PROC HPFEVENTS *options*;
  VAR *variables*;
  BY *variables*;
  ID *variable* **INTERVAL=** *interval options*;
  EVENTDEF *variable= do-list / options*;
  EVENTKEY *<variable=> predefined-event-keyword /options*;
  EVENTCOMB *variable= variable-list / options*;
  EVENTGROUP *<variable=> predefined-event-keyword* ;
  EVENTGROUP *variable= variable-list* ;
  EVENTDATA *options*;
  EVENTDUMMY *options*;

Functional Summary

The statements and options controlling the HPFEVENTS procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify event combination	EVENTCOMB	
specify event definition	EVENTDEF	
specify group of events	EVENTGROUP	
use predefined event definition	EVENTKEY	
specify event data set	EVENTDATA	
specify dummy data set	EVENTDUMMY	
specify the time ID variable	ID	
specify variables to be copied to the dummy data set	VAR	
Data Set Options		
specify the input data set	PROC HPFEVENTS	DATA=
specify an events input data set	EVENTDATA	IN=
specify an events output data set	EVENTDATA	OUT=
specify that the events output data set will be condensed	EVENTDATA	CONDENSE
specify a dummy output data set	EVENTDUMMY	OUT=
specify starting time ID value	ID	START=
specify ending time ID value	ID	END=

Description	Statement	Option
Dummy Variable Format Options		
extend dummy variables past end of series	PROC HPFEVENTS	LEAD=
specify missing value interpretation	ID	SETMISSING=
specify frequency of the dummy variable(s)	ID	INTERVAL=
specify interval alignment	ID	ALIGN=
Miscellaneous Options		
specify that variables in output data sets are in sorted order	PROC HPFEVENTS	SORTNAMES
limit error and warning messages	PROC HPFEVENTS	MAXERROR=

PROC HPFEVENTS Statement

PROC HPFEVENTS *options;*

The following options can be used in the PROC HPFEVENTS statement.

DATA= *SAS-data-set*
names the SAS data set containing the variables used in the VAR, ID, and BY statements. If the DATA= option is not specified, the most recently created SAS data set is used.

LEAD= *n*
specifies the number of periods to extend the dummy variable beyond the time series. The default is LEAD=0.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series. Thus, if a series has missing values at the end, the actual number of dummy values beyond the last nonmissing value will be greater than the LEAD= value.

MAXERROR= *number*
limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERROR=25. This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

SORTNAMES
specifies that the events and variables in the output data sets are printed in alphabetical order.

BY Statement

BY *variables;*

A BY statement can be used with PROC HPFEVENTS to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

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 actual time series. The information specified affects all dummy variables output using the EVENTDUMMY statements. If no dummy variables are requested, the ID statement has no impact on processing, since the EVENTDEF definitions are independent of the time identification values of a time series. 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. When the observation number is used as the time ID, only EVENT timing values that are based on observation numbers are applied to the time series to create dummy variables; timing values based on SAS date or datetime values are ignored.

The following options can be used with the ID statement.

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 option and the START= option can be used to ensure that data associated with 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 input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. Refer to *SAS/ETS User's Guide* for the intervals that can be specified.

SETMISSING= *option | number*
specifies how missing values are assigned in the time series copied to the dummy

data set when there is no observation matching the time ID in the input data set. 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.
SKIP	If the observation for the time ID value is missing in the input data set, then the corresponding observation is skipped in the dummy data set. This option may be useful if dummies are to be used as predictor values and you wish apply the PROC HPFENGINE ACCUMULATION option to the dummy data.

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 option and the END= option can be used to ensure that data associated with each by group contains the same number of observations.

EVENTDEF Statement

EVENTDEF *SAS-variable-name= timing-value-list / qualifier options;*

An EVENTDEF statement can be used with PROC HPFEVENTS to define an event. Although an event occurs at one or more time values, the event definition is independent of the time ID; that is, the event is a function that operates on a time ID variable. Once defined, the event can be output using the OUT= option of the EVENTDATA statement. A dummy variable for the event can be output using the OUT= option of the EVENTDUMMY statement. The dummy variable is created by evaluating the event with respect to the time ID. If a time ID is not specified using the ID statement, then the BY-group observation number is used as the time ID. More than one EVENTDEF statement can be specified.

Once defined, an event is referenced using its SAS-variable-name. When the event is output using the EVENTDATA statement, the event is identified by its SAS-variable-name. When a dummy is created using the event definition, the dummy variable name is the same as the event SAS-variable-name.

Each event must have a unique SAS-variable-name. If two event definitions have the same name, the following rules apply. If two EVENTDEF statements exist using the same name, the later statement is used. If an event is defined in both an EVENTDEF statement and in a data set specified using the EVENTDATA statement, the definition in the EVENTDEF statement is used. Any event defined using an EVENTDEF or EVENTDATA statement is used rather than a SAS predefined event.

Each EVENTDEF statement must be defined using one or more event timing values. The timing values may be specified using a list. Each item in the list may be a SAS date keyword, an integer, a SAS-date, a SAS-datetime, or a do-list. For example, the EVENTDEF statement below specifies timing values using each of these methods in the order listed.

```
EVENTDEF EVENT1= USINDEPENDENCE 10 '25Dec2000'D
                 '01Mar1990:15:03:00'DT
                 '01Jan2000'D to '01Mar2000'D by month;
```

The timing values are interpreted as: any July 4 in the series; the 10th observation; December 25, 2000; March 1, 1990 at 3:03PM; January 1, 2000; February 1, 2000; and March 1, 2000. The following two EVENTDEF statements specify identical timing values.

```
EVENTDEF MYFIRSTEVENT= '01Jan2000'D to '01Mar2000'D by month;
EVENTDEF MYNEXTEVENT= ( '01Jan2000'D, '01Feb2000'D, '01Mar2000'D );
```

The timing value list can be enclosed in parentheses, and commas can separate the items in the list. Numbers are always interpreted as observation numbers. The do-list may be based on observation numbers, SAS-dates, or SAS-datetimes. However, the first and second values in the list must be of the same type. The SAS grammar always expects the type of the second value to be the same as the type of the first value, and tries to interpret the statement in that fashion. The following statement yields erratic results.

```
EVENTDEF BADEVENT= '01Jan2000'D to '01Mar2000:00:00:00'DT by month;
```

Either the HPFEVENTS procedure produces a list much longer than expected or the procedure does not have enough memory to execute. **Note:** You should never mix date, datetime, and integer types in a do-list.

Table 12.1 shows the date keywords that can be used in a timing value list and their definitions.

Table 12.1. Holiday Date Keywords and Definitions

Date Keyword	**Definition**
EASTER	Easter Sunday
THANKSGIVING	4th Thursday in November
BOXING	December 26th
CANADA	July 1st
CHRISTMAS	December 25th
FATHERS	3rd Sunday in June
HALLOWEEN	October 31st
USINDEPENDENCE	July 4th
LABOR	1st Monday in September
MEMORIAL	last Monday in May
MOTHERS	2nd Sunday in May
NEWYEAR	January 1st
THANKSGIVINGCANADA	2nd Monday in October
VALENTINES	February 14th
VICTORIA	Monday on or preceding May 24th
CANADAOBSERVED	July 1st, or July 2nd, if July 1st is a Sunday

The date of Easter is calculated using a method described by Montes (2001b).

Table 12.2 shows the seasonal date keywords that can be used in a timing value list and their definitions.

Table 12.2. Seasonal Date Keywords and Definitions

Date Keyword	**Definition**
SECOND_1, ... SECOND_60	The specified second.
MINUTE_1, ... MINUTE_60	The beginning of the specified minute.
HOUR_1, ... HOUR_24	The beginning of the specified hour.
SUNDAY, ... SATURDAY	All SUNDAYs, etc., in the time series.
WEEK_1, ... WEEK_53	The first day of the nth week of the year. PULSE=WEEK.n shifts this date for $n \neq 1$.
TENDAY_1, ... TENDAY_36	The 1st, 11th, or 21st of the appropriate month.
SEMIMONTH_1, ... SEMIMONTH_24	The 1st or 16th of the appropriate month.
JANUARY, ... DECEMBER	The 1st of the specified month.
QTR_1, QTR_2, QTR_3, QTR_4	The first date of the quarter. PULSE=QTR.n shifts this date for $n \neq 1$.
SEMIYEAR_1, SEMIYEAR_2	The first date of the semiyear. PULSE=SEMIYEAR.n shifts this date for $n \neq 1$.

When dummies are created, each timing value is evaluated with respect to the time ID. You should take care to choose the event timing value(s) that are consistent with the time ID. In particular, date and datetime timing value(s) are ignored when the time ID is based on the observation number.

The qualifier options define a function to be applied at each timing value. The following qualifier options can be used with the EVENTDEF statement.

TYPE= *option*

specifies the type of event variable. Each type uses a different formula to create the dummy variables. The formula for each TYPE= option is dependent on the other qualifiers that are specified in the EVENTDEF options. The formula is applied to each timing value specified in the timing-value list. The TYPE= option accepts the following values: POINT | LS | RAMP | TR | TEMPRAMP | TC | LIN | LINEAR | QUAD | CUBIC | INV | INVERSE | LOG | LOGARITHMIC. Table 12.6 illustrates the basic shape for each TYPE= value. POINT is the default type.

Table 12.3, Table 12.4, and Table 12.5 show the formulas used to calculate the dummy variables for the event. Table 12.4 shows the formula used for each type of event when the event extends infinitely both before and after the timing value. Table 12.5 shows the formula used for each type of event for finite duration values. In the formulas, t_i is the observation specified by the ith timing value in the timing value list, VALUE=ν, TCPARM=ϕ, AFTER=(DURATION=n SLOPE=s_a), BEFORE=(DURATION=m SLOPE=s_b), and PULSE=$interval$. Table 12.3 shows how to calculate t_b and t_e, which are based on the DURATION= values. t_b and t_e are the beginning and ending observations of the event definition. (For TYPE=RAMP, the ramp persists past the top of the ramp.) For more information on matching SAS date values to observations, refer to Chapter 2, “Working with Time Series Data” (*SAS/ETS User’s Guide*) and Chapter 3, “Date Intervals, Formats, and Functions” (*SAS/ETS User’s Guide*).

When one DURATION= value is finite, and the other is infinite, this is equivalent to extending the finite portion of the event infinitely in one direction. This principle may be understood by examining the results of the following EVENTDEF statements:

```
eventdef monlygg= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4);
eventdef minfgg= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4)
         AFTER=(SLOPE=GROWTH DURATION=ALL) ;
eventdef minfgd= '01Jun1951'D  /  TYPE=RAMP
         BEFORE=(SLOPE=GROWTH DURATION=4)
         AFTER=(SLOPE=DECAY DURATION=ALL) ;
```

In the section “Examples” on page 438, Example 12.5 shows how PROC HPFEVENTS interprets each of these statements.

Table 12.3. Calculating the Beginning and Ending Observation for Events

BEFORE= (DURATION=value)	**PULSE=value**	**Definition of t_b**
ALL	N/A	$t_b = 1$, the first observation in the data set or $t_b =$ the observation specified by START=
$m = 0$	not specified	$t_b = t_i$ (the observation specified by the timing value)
$m > 0$	not specified	$t_b = t_i - m$
$m \geq 0$	*interval*	$t_b =$ the observation specified by the date INTNX(*interval*, timing value, -*m*, ‘begin’)
AFTER= (DURATION=value)	**PULSE=value**	**Definition of t_e**
ALL	N/A	$t_e =$ the last observation in the data set or $t_e =$ the observation specified by END=
$n = 0$	not specified	$t_e = t_i$, the observation specified by the timing value
$n > 0$	not specified	$t_e = t_i + n$
$n \geq 0$	*interval*	$t_e =$ the observation specified by the date INTNX(*interval*, timing value, *n*, ‘end’)

Table 12.4. Event Types for Infinite Durations (m =ALL and n =ALL)

Type	**Description**	**Definition**
POINT	point or pulse	$\xi_{it} = \nu$, for all t
LS	level shift	$\xi_{it} = \nu$, for all t
RAMP	ramp	
	SLOPE=GROWTH	$\xi_{it} = \nu(t - t_i)$, for all t
	SLOPE=DECAY	$\xi_{it} = \nu(t_i - t)$, for all t
	$s_b =$ GROWTH $s_a =$ DECAY	$\xi_{it} = \nu(t - t_i)$, if $t \leq t_i$ $\xi_{it} = \nu(t_i - t)$, if $t_i \leq t$
	$s_b =$ DECAY $s_a =$ GROWTH	$\xi_{it} = \nu(t_i - t)$, if $t \leq t_i$ $\xi_{it} = \nu(t - t_i)$, if $t_i \leq t$
TEMPRAMP or TR	temporary ramp	TEMPRAMP is the same as RAMP for infinite cases
TC	temporary change	
	SLOPE=GROWTH	$\xi_{it} = \nu\phi^{(t_i - t)}$, for all t

Table 12.4. (continued)

Type	Description	Definition
	SLOPE=DECAY	$\xi_{it} = \nu\phi^{(t-t_i)}$, for all t
	s_b = GROWTH s_a = DECAY	$\xi_{it} = \nu\phi^{(t_i-t)}$, if $t \le t_i$ $\xi_{it} = \nu\phi^{(t-t_i)}$, if $t_i \le t$
	s_b = DECAY s_a = GROWTH	$\xi_{it} = \nu\phi^{(t-t_i)}$, if $t \le t_i$ $\xi_{it} = \nu\phi^{(t_i-t)}$, if $t_i \le t$
LINEAR or LIN	linear trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)$, for all t
QUAD	quadratic trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)^2$, for all t
CUBIC	cubic trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)^3$, for all t
INVERSE or INV	inverse trend SLOPE= does not apply	$\xi_{it} = \nu/t$, for all t
LOGARITHMIC or LOG	log trend SLOPE= does not apply	$\xi_{it} = \nu \log(t)$, for all t

Table 12.5. Event Types (for finite m,n)

Type	Description	Definition
POINT	point or pulse	$\xi_{it} = \nu$, if $t_b \le t \le t_e$ ξ_{it} = undefined, otherwise
LS	level shift	$\xi_{it} = \nu$, if $t_b \le t \le t_e$ ξ_{it} = undefined, otherwise
RAMP	ramp $m = n = 0$ PULSE=*interval* $\le$ width of an observation	$\xi_{it} = 0$, if $t = t_i$ ξ_{it} = undefined, otherwise
	SLOPE=GROWTH	$\xi_{it} = \nu(t - t_b)/(t_e - t_b)$, if $t_b \le t \le t_e$

Table 12.5. (continued)

Type	Description	Definition
		$\xi_{it} = \nu$, if $t > t_e$ $\xi_{it} =$ undefined, otherwise
	SLOPE=DECAY	$\xi_{it} = \nu$, if $t < t_b$ $\xi_{it} = \nu(t_e - t)/(t_e - t_b)$, if $t_b \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	$s_b =$ GROWTH $s_a =$ DECAY $m > 0, n > 0$	$\xi_{it} = \nu(t - t_b)/(t_i - t_b)$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu(t_e - t)/(t_e - t_i)$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	$s_b =$ DECAY $s_a =$ GROWTH $m > 0, n > 0$	$\xi_{it} = \nu$, if $t < t_b$ $\xi_{it} = \nu(t_i - t)/(t_i - t_b)$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu(t - t_i)/(t_e - t_i)$, if $t_i \leq t \leq t_e$ $\xi_{it} = \nu$, if $t > t_e$
TEMPRAMP or TR	temporary ramp $m = n = 0$ PULSE=*interval* $\leq$ width of an observation	$\xi_{it} = 0$, if $t = t_i$ $\xi_{it} =$ undefined, otherwise
	SLOPE=GROWTH	$\xi_{it} = \nu(t - t_b)/(t_e - t_b)$, if $t_b \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	SLOPE=DECAY	$\xi_{it} = \nu(t_e - t)/(t_e - t_b)$, if $t_b \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	$s_b =$ GROWTH $s_a =$ DECAY $m > 0, n > 0$	$\xi_{it} = \nu(t - t_b)/(t_i - t_b)$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu(t_e - t)/(t_e - t_i)$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	$s_b =$ DECAY $s_a =$ GROWTH $m > 0, n > 0$	$\xi_{it} = \nu(t_i - t)/(t_i - t_b)$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu(t - t_i)/(t_e - t_i)$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
TC	temporary change SLOPE=GROWTH	$\xi_{it} = \nu\phi^{(t_e - t)}$, if $t_b \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	SLOPE=DECAY	$\xi_{it} = \nu\phi^{(t - t_b)}$, if $t_b \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	$s_b =$ GROWTH $s_a =$ DECAY $m > 0, n > 0$	$\xi_{it} = \nu\phi^{(t_i - t)}$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu\phi^{(t - t_i)}$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise

Table 12.5. (continued)

Type	Description	Definition
	s_b = DECAY s_a = GROWTH $0 < m \leq n$	$\xi_{it} = \nu\phi^{((t_e - t_i)+(t-t_i))}$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu\phi^{(t_e - t)}$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
	s_b = DECAY s_a = GROWTH $0 < n \leq m$	$\xi_{it} = \nu\phi^{(t - t_b)}$, if $t_b \leq t \leq t_i$ $\xi_{it} = \nu\phi^{((t_i - t_b)+(t_i - t))}$, if $t_i \leq t \leq t_e$ $\xi_{it} =$ undefined, otherwise
LINEAR or LIN	linear trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)$, if $t_b \leq t \leq t_e$
QUAD	quadratic trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)^2$, if $t_b \leq t \leq t_e$
CUBIC	cubic trend SLOPE= does not apply	$\xi_{it} = \nu(t - t_i)^3$, if $t_b \leq t \leq t_e$
INVERSE or INV	inverse trend SLOPE= does not apply	$\xi_{it} = \nu/(t - t_b + 1)$, if $t_b \leq t \leq t_e$
LOGARITHMIC or LOG	log trend SLOPE= does not apply	$\xi_{it} = \nu \log(t - t_b + 1)$, if $t_b \leq t \leq t_e$

Note that undefined values are set to zero after all timing values have been evaluated. See the RULE= option for details on evaluating overlapping timing values.

VALUE= *number*

specifies the event indicator value, ν. The default event indicator value is one ($\nu = 1.0$). Table 12.5 provides details about the effect of the event indicator value on the dummy variables. However, for TYPE=POINT | LS | RAMP | TR | TC events consisting of a single timing value with finite duration, the user can think of the event indicator value as the maximum amplitude: the values of the dummy should be bounded below by zero and above by the event indicator value. For trend events (TYPE = LINEAR | QUAD | CUBIC | INV | LOG), the event indicator value is the coefficient of the term.

SHIFT= *number*

specifies the number of pulses to shift the timing value, δ. The default is not to shift the timing value ($\delta = 0$). When the SHIFT= option is used, all timing values in the list, including those generated by date keywords are shifted. Thus, SHIFT= can be used with EASTER to specify Ecclesiastical Holidays that are based on Easter. For example,

```
EVENTDEF GoodFriday= EASTER / SHIFT=-2 PULSE=DAY;
```

specifies Good Friday, which is defined as 2 days before Easter (Montes 2001a).

PULSE= *interval*

specifies the interval to be used with the DURATION= option to determine the width of the event. The default pulse is one observation. When no DURATION= values are specified, and the PULSE= option is specified, the DURATION= values are set to zero. Refer to *SAS/ETS User's Guide* for the intervals that can be specified.

BEFORE= *(option-list)*
AFTER= *(option-list)*

specifies options that control the event definition before and after the timing value. The DURATION= and SLOPE= options are used within the parentheses in the BEFORE=( ) and AFTER=( ) options. The SLOPE= option is ignored if the corresponding DURATION=0.

DURATION= *number*

specifies the event duration before the timing value when used in the BEFORE=( ) option or after the timing value when used in the AFTER=( ) option. The event always occurs at the timing value. You would specify 3 observations before, 1 observation at the timing value, and 4 after the timing value for a total of $3+1+4 = 8$ observations as follows:

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3) AFTER=(DURATION=4);
```

You would specify 3 weeks before, the week of the timing value, and 4 weeks after the timing value using a combination of the BEFORE=, AFTER=, and PULSE= options as follows:

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3) AFTER=(DURATION=4)
                            PULSE=WEEK;
```

DURATION=ALL implies that the event should be extended to the beginning (BEFORE=) or end (AFTER=) of the series. If only one DURATION= value is specified, the other value is assumed to be zero. When neither DURATION= value is specified, and the PULSE= value is specified, both DURATION= values are set to zero. When neither DURATION= value is specified, and the PULSE= value is not specified, then both DURATION= values are assigned default values based on the TYPE= option. For polynomial trend events (TYPE = LINEAR | QUAD | CUBIC), the default DURATION= value is ALL for both the BEFORE=( ) option and the AFTER=( ) option. For other events, the default value for the BEFORE=( ) option is always zero, and the default event duration for the AFTER=( ) option depends on the TYPE= option. Table 12.6 shows default duration values by TYPE= value and how the basic event shape depends on the duration value. DURATION=ALL is represented in the event definition data set as a special missing value displayed as “A”. For more information on special missing values, refer to SAS System Concepts in *SAS Language Reference: Concepts*.

Table 12.6. Default DURATION= values When No DURATION= value nor PULSE= value Is Specified

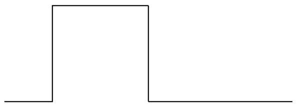
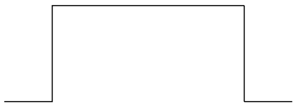
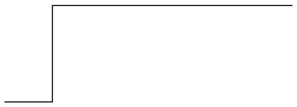
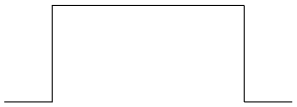
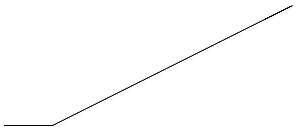
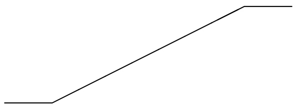
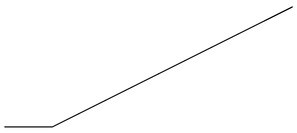
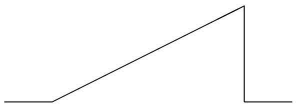
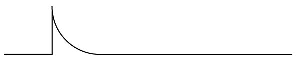
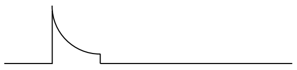
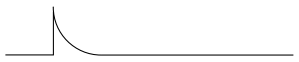
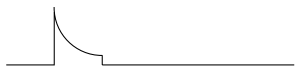
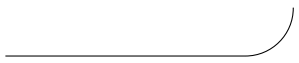
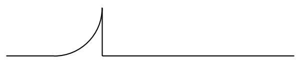
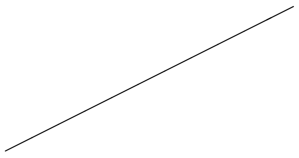
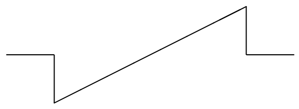
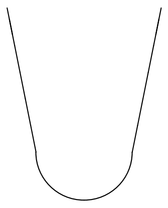
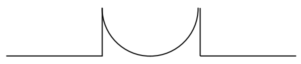
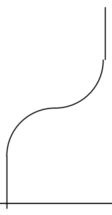
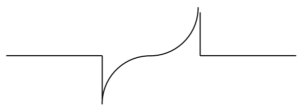
Non-trend TYPE= (BEFORE= Default is 0)	**AFTER= (DURATION=) Default**	**Default Shape**	**Shape when finite AFTER= duration > 0**
TYPE=POINT	default is zero		
TYPE=LS	default is ALL (or end of series)		
TYPE=RAMP	default is ALL (or end of series)		
TYPE= TEMPRAMP or TR	default is ALL (or end of series)		
TYPE=TC	default is ALL (or end of series)		

Table 12.6. (continued)

Non-trend TYPE= (BEFORE= Default is 0)	AFTER= (DURATION=) Default	Default Shape	Shape when finite AFTER= duration > 0
TYPE=INV or INVERSE	default is ALL (or end of series)		
TYPE=LOG or LOGARITHMIC	default is ALL (or end of series)		
Trend TYPE=	**BEFORE= and AFTER= (DURATION=) Default**	**Default Shape**	**Shape when finite BEFORE= and AFTER= duration > 0**
TYPE=LINEAR TYPE=LIN	default is ALL (or entire series)		
TYPE=QUAD	default is ALL (or entire series)		
TYPE=CUBIC	default is ALL (or entire series)		

SLOPE= *option*

specifies whether a ramp or temporary change type is growth or decay. SLOPE is ignored unless TYPE=RAMP, TYPE=TR, TYPE=TEMPRAMP, or TYPE=TC. SLOPE= is also ignored if the corresponding DURATION=0. The SLOPE= value in the BEFORE= option controls the slope before the timing value and the SLOPE= value in the AFTER= option controls the slope after the timing value. The SLOPE= option accepts the values: GROWTH | DECAY. GROWTH is the default in all cases except TYPE=TC. For TYPE=TC, the default is BEFORE=(SLOPE=GROWTH) and AFTER=(SLOPE=DECAY). The following statement

```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3 SLOPE=GROWTH)
                            AFTER=(DURATION=4) SLOPE=DECAY)
                            TYPE=RAMP;
```

specifies a ramp up, followed by a ramp down. The event dummy observations immediately preceding the timing value contain the following values: 0, $\frac{1}{3}$, $\frac{2}{3}$. The observation at the timing value has a value of 1. The observations immediately after the timing value are $\frac{3}{4}$, $\frac{2}{4}$, $\frac{1}{4}$, 0.

TCPARM= *number*

specifies a parameter $0 \le \phi \le 1$ used in the growth/decay equation for TYPE=TC given in Table 12.5. The TCPARM= value is the rate of growth or decay. A larger TCPARM= value causes faster growth or decay. TCPARM is ignored unless TYPE=TC. The default value is 0.5.

RULE=

specifies the action to take when the defined event has multiple timing values which overlap. When the timing values do not overlap, RULE= has no impact since if there is only one defined value for an observation, that value is always used. The RULE= option accepts the following values: ADD | MAX | MIN | MINNZ | MINMAG | MULT . ADD is the default. Table 12.7 explains how the RULE= option is interpreted when the value for an observation is defined by multiple timing values. Because the range of the event associated with a timing value might not include the all the observations in the series, RULE= might be interpreted differently when using multiple timing values in one EVENTDEF statement versus defining a combination event using the EVENTCOMB statement. Thus, the dummy variables TWOTIMING and TWOEVENTS defined in the following statements are different:

```
eventdef xmasrp= CHRISTMAS / BEFORE=(SLOPE=GROWTH DURATION=3)
                             TYPE=RAMP RULE=MIN ;
eventdef easterrp= EASTER / BEFORE=(SLOPE=GROWTH DURATION=3)
                            TYPE=RAMP RULE=MIN ;
eventdef twotiming= EASTER CHRISTMAS /
                            BEFORE=(SLOPE=GROWTH DURATION=3)
                            TYPE=RAMP RULE=MIN ;
eventcomb twoevents= easterrp xmasrp / RULE=MIN ;
```

In the section "Examples" on page 438, Example 12.1 shows how PROC HPFEVENTS interprets each of these statements.

Table 12.7. Definition of RULE= Values

RULE	Name	Definition
ADD	Add	Add the values.
MAX	Maximum	Use the maximum value.
MIN	Minimum	Use the minimum value.
MINNZ	Minimum Nonzero	Use the minimum nonzero value.
MINMAG	Minimum Magnitude	Use the value whose magnitude is the least.
MULT	Multiply	Multiply the values.

PERIOD= *interval*

specifies the interval for the frequency of the event. For example, PERIOD=YEAR should produce a dummy value that is periodic in a yearly pattern. If the PERIOD= option is omitted, the event is not periodic. The PERIOD= option also does not apply to observation numbers, which are not periodic, or to date keywords, which have their own periodicity. Refer to *SAS/ETS User's Guide* for the intervals that can be specified.

LABEL= *'SAS-label'*

specifies a label for the dummy variable for this event. 'SAS-label' is a quoted text string of up to 256 characters. The default label is "Dummy Variable for Event <variable-name>" where <variable-name> is the name specified in the EVENT statement. The label is also stored as a description in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set, but the default label is still used for the dummy variable.

EVENTKEY Statement

EVENTKEY *<variable=> predefined-event-keyword /options;*

EVENTKEY *<variable=> user-defined-event-keyword /options;*

An EVENTKEY statement can be used to alter a user defined simple event or a SAS predefined event or to create a new event based on a user defined simple event or a SAS predefined event.

An EVENTKEY statement can be used with PROC HPFEVENTS to make a SAS predefined event available for processing. The EVENTKEY statement constructs a SAS simple EVENT for each SAS predefined event keyword. The SAS predefined events are also available directly through PROC HPFDIAGNOSE and PROC HPFENGINE. Each EVENTKEY variable has a predefined set of timing values and qualifiers associated with the predefined event keyword. The options are the same as in the EVENTDEF statement and can be used to redefine the qualifiers associated with the predefined event. As shown in the "Getting Started" section on page 410, the default SAS variable name for the predefined event is the predefined event keyword. However, the user can specify a SAS variable name for the event. For example, you can rename the CHRISTMAS predefined EVENT to XMAS using the following statement:

```
EVENTKEY XMAS= CHRISTMAS;
```

If the user redefines the qualifiers associated with a SAS predefined EVENT and does not rename the event, then that has the impact of redefining the SAS predefined event, since any user definition takes precedence over a SAS predefined definition. The following example produces an event FALLHOLIDAYS with a pulse of 1 day at Halloween and a pulse of 1 month at Thanksgiving.

```
EVENTKEY THANKSGIVING / PULSE=MONTH;
EVENTCOMB FALLHOLIDAYS= HALLOWEEN THANKSGIVING;
```

SAS predefined events are based on either a SAS date keyword or an additive outlier or level shift based on a timing value. Table 12.8 describes how to construct a SAS predefined event keyword. It also gives the default qualifier options for those predefined events.

An EVENTKEY statement may be used in a similar manner to modify or clone a user defined simple event. In the following example, the EVENTDEF statement is used to define a simple event named SPRING. The EVENTKEY statement is used to modify the SPRING event definition, and then the EVENTKEY statement is used to create a new event named SPRINGBREAK based on the previously defined user event named SPRING. So the example defines a total of two events, SPRING and SPRINGBREAK. The EVENTKEY statement may be used to modify the qualifiers; it may not be used to modify the timing value(s).

```
EVENTDEF SPRING = '20MAR2005'D;
EVENTKEY SPRING / PULSE=DAY;
EVENTKEY SPRINGBREAK = SPRING / PULSE=WEEK;
```

Suppose that the events above are stored in a data set named SPRINGHOLIDAYS. The first EVENTKEY statement in the example below would clone SPRING as an event named FirstDayOfSpring. The second EVENTKEY statement will change the case of the SPRINGBREAK event name.

```
EVENTDATA IN=SPRINGHOLIDAYS;
EVENTKEY FirstDayOfSpring = SPRING;
EVENTKEY SpringBreak = springbreak;
```

Event names that refer to a previously defined event are not case sensitive. However, event names that are used to create a new event will have the case preserved in the _NAME_ variable of the EVENTDATA OUT= data set and the variable name used in the EVENTDUMMY OUT= data set.

Table 12.8. Definitions for EVENTKEY Predefined Event Keywords

Variable Name or Variable Name Format	Description	Qualifier Options

AO<obs>OBS
AO<date>D
AO<datetime>DT
	Outlier	TYPE=POINT VALUE=1
		BEFORE=(DURATION=0)
		AFTER=(DURATION=0)

LS<obs>OBS
LS<date>D
LS<datetime>DT
	Level Shift	TYPE=LS VALUE=1
		BEFORE=(DURATION=0)
		AFTER=(DURATION=ALL)

<date keyword>	Date Pulse	TYPE=POINT VALUE=1
		BEFORE=(DURATION=0)
		AFTER=(DURATION=0)
		PULSE=DAY

LINEAR
QUAD
CUBIC
	Polynomial Trends	TYPE=LIN
		TYPE=QUAD
		TYPE=CUBIC
		VALUE=1 BEFORE=(DURATION=ALL)
		AFTER=(DURATION=ALL)
		default timing value is 0 observation

INVERSE
LOG
	Trends	TYPE=INV
		TYPE=LOG
		VALUE=1 BEFORE=(DURATION=0)
		AFTER=(DURATION=ALL)
		default timing value is 0 observation

<seasonal keywords>	Seasonal	TYPE=POINT
		PULSE= depends on keyword
		VALUE=1 BEFORE=(DURATION=0)
		AFTER=(DURATION=0)
		timing values based on keyword

The date keywords described in Table 12.1 in the section on the EVENTDEF statement can be used as SAS predefined event keywords. The timing value(s) are as defined in Table 12.1 and the default qualifiers are as shown in Table 12.8. The seasonal keywords described in Table 12.2 in the section on the EVENTDEF statement shows the seasonal keywords that can be used as SAS predefined event keywords. The default qualifiers for seasonal keywords are shown in Table 12.8. Table 12.9 gives a more detailed description of how date and observation numbers are encoded into AO and LS type predefined events.

Table 12.9. Details for Encoding Date Information into AO and LS EVENTKEY Variable Names

Variable Name Format	Example	Refers to
AO<int>OBS	AO15OBS	15th observation
AO<date>D	AO01JAN2000D	'01JAN2000'D
AO<date>h<hr>m<min>s<sec>DT	AO01Jan2000h12m34s56DT	'01Jan2000:12:34:56'DT
LS<int>OBS	LS15OBS	15th observation
LS<date>D	LS01JAN2000D	'01JAN2000'D
LS<date>h<hr>m<min>s<sec>DT	LS01Jan2000h12m34s56DT	'01Jan2000:12:34:56'DT

EVENTCOMB Statement

EVENTCOMB *variable= variable-list /options;*

An EVENTCOMB statement can be used with PROC HPFEVENTS to create a new event from one or more events that have previously been defined.

The following options can be used with the EVENTCOMB statement.

RULE=

specifies the action to take when combining events. The RULE= option accepts the following values: ADD | MAX | MIN | MINNZ | MINMAG | MULT . ADD | MAX | MIN | MINNZ | MINMAG . ADD is the default. Table 12.7 explains how the RULE= option is interpreted. Example 12.1 shows how PROC HPFEVENTS interprets the RULE= option in the EVENTDEF and EVENTCOMB statements.

LABEL= *'SAS-label'*

specifies a label for the dummy variable for this event. 'SAS-label' is a quoted text string of up to 256 characters. The default label is "Dummy Variable for Event <variable-name>" where <variable-name> is the name specified in the EVENT statement. The label is also stored as a description in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set, but the default label is still used for the dummy variable.

EVENTGROUP Statement

EVENTGROUP *<variable=> predefined-eventgroup-keyword ;*

EVENTGROUP *variable= (variable-list) ;*

An EVENTGROUP statement can be used with PROC HPFEVENTS to create an event group or make a predefined event group available for processing. The EVENTGROUP statement constructs a SAS complex EVENT. A complex EVENT is an event that is represented by multiple dummy variables. For example, seasonal effects usually require multiple dummy variables. The SAS predefined event groups are also available directly through PROC HPFENGINE. Each EVENTGROUP predefined group keyword has a predefined set of event keywords associated with the predefined group. The default SAS variable name for the predefined event is the predefined event keyword. However, the user can specify a SAS variable name for the event. For example, you can rename the DAYS predefined EVENT group to TD using the following statement:

```
   EVENTGROUP TD= DAYS;
```

The following option can be used with the EVENTGROUP statement.

LABEL= *'SAS-label'*

specifies a description which is stored in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set.

Table 12.10 describes the SAS predefined event group keywords. The SEASONAL group is a PREDEFINED COMPLEX EVENT; the SEASONAL group is interpreted to be one of the other SEASONAL groups at the time that dummy variables are created based on the ID statement. The ID statement could be the ID statement associated with either PROC HPFEVENTS or PROC HPFENGINE.

Table 12.10. Definitions for EVENTGROUP Predefined Event Group Keywords

Variable Name	**Description**	**Associated Event Keywords**
Seasonal	Seasonal	Depending on ID statement:
		SECOND_1, ... SECOND_60 or
		MINUTE_1, ... MINUTE_60 or
		HOUR_1, ... HOUR_24 or
		SUNDAY, ... SATURDAY or
		WEEK_1, ... WEEK_53 or
		TENDAY_1, ... TENDAY_36 or
		SEMIMONTH_1, ... SEMIMONTH_24 or
		JANUARY, ... DECEMBER or
		QTR_1, QTR_2, QTR_3, QTR_4 or
		SEMIYEAR_1, SEMIYEAR_2
SECONDS	Seasonal	SECOND_1, ... SECOND_60
MINUTES	Seasonal	MINUTE_1, ... MINUTE_60
HOURS	Seasonal	HOUR_1, ... HOUR_24
DAYS	Seasonal	SUNDAY, ... SATURDAY
WEEKDAYS	Seasonal	MONDAY, ... FRIDAY
		(FRIDAY includes SATURDAY and SUNDAY)
WEEKS	Seasonal	WEEK_1, ... WEEK_53
TENDAYS	Seasonal	TENDAY_1, ... TENDAY_36
SEMIMONTHS	Seasonal	SEMIMONTH_1, ... SEMIMONTH_24
MONTHS	Seasonal	JANUARY, ... DECEMBER
QTRS	Seasonal	QTR_1, QTR_2, QTR_3, QTR_4
SEMIYEARS	Seasonal	SEMIYEAR_1, SEMIYEAR_2
CUBICTREND	Trend	LINEAR, QUAD, CUBIC
QUADTREND	Trend	LINEAR, QUAD

Table 12.8 gives more detail on the seasonal and trend SAS predefined EVENTS that compose the EVENT groups.

EVENTDATA Statement

EVENTDATA *options;*

An EVENTDATA statement can be used with PROC HPFEVENTS to input events from an events data set and to output events to an events data set. Either the IN= or the OUT= option must be specified.

The following options can be used with the EVENTDATA statement.

IN= *SAS-data-set*
names an input data set that contains event definitions to be used in the PROC HPFEVENTS procedure.

OUT= *SAS-data-set*
names the output data set to contain the event definitions as specified in the EVENTDATA IN= data sets and the EVENTDEF, EVENTKEY, and EVENTCOMB statements. The OUT= data set can then be used in other SAS procedures to define events.

CONDENSE
specifies that the EVENTDATA OUT= data set is condensed; any variables that contain only default values are omitted from the data set. The EVENTDATA IN= option reads both condensed data sets and data sets that have not been condensed. For more details, see the "EVENTDATA OUT= Data Set" section on page 436.

EVENTDUMMY Statement

EVENTDUMMY *options;*

An EVENTDUMMY statement can be used with PROC HPFEVENTS to output dummy variables for events to a data set. The OUT= option must be specified.

The following option can be used with the EVENTDUMMY statement.

OUT= *SAS-data-set*
names the output data set to contain the dummy variables for the specified events based on the ID information as specified in the ID statement. The OUT= data set also includes variables as specified in the VAR, BY, and ID statements.

VAR Statement

VAR *variables;*

A VAR statement can be used with PROC HPFEVENTS to copy input variables to the output dummy data set. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are selected except those appearing in a BY or ID statement.

Details

Missing Value Interpretation

When the EVENTDUMMY statement is used to create dummy variables, you may need to specify the handling of missing observations in the input data set, that is, where the observation corresponding to a time ID is missing from the data set. In that case, the input data set does not contain a value for the variables to be copied to the EVENTDUMMY OUT= data set. Sometimes missing values should be interpreted as unknown values. The forecasting models used by the HPFENGINE procedure can effectively handle missing values (see Chapter 10, “The HPFENGINE Procedure”). In this case, SETMISSING=MISSING can be used. But sometimes missing values are known, such as when no observations should be interpreted as no (zero) value. In this case, the SETMISSING=0 option should be used. In other cases, missing time IDs should be skipped, such as when the data is to be accumulated at a later time. In this case, SETMISSING=SKIP should be used.

Data Set Output

The HPFEVENTS procedure can create the EVENTDATA OUT= and EVENTDUMMY OUT= data sets. The EVENTDATA OUT= data set contains the EVENT definitions that may be used for input to another SAS procedure. The EVENTDUMMY OUT= data set contains the variables listed in the BY statement, the ID variable, any variables defined by the VAR statement, and any dummy variables generated by the procedure.

EVENTDATA OUT= Data Set

The EVENTDATA OUT= data set contains the variables listed below. The default values for the CONDENSE option are also given. When all the observations in the variable are equal to the default value, the variable can be omitted from the event definition data set.

NAME	EVENT variable name. _NAME_ is displayed with the case preserved. Since _NAME_ is a SAS variable name, the event may be referenced using any case. The _NAME_ variable is required; there is no default.
CLASS	Class of EVENT: SIMPLE, COMBINATION, PREDEFINED. The default for _CLASS_ is SIMPLE.
KEYNAME	Contains either a date keyword (SIMPLE EVENT) or a predefined EVENT variable name (PREDEFINED EVENT) or an event name (COMBINATION event). All _KEYNAME_ values are displayed in upper case. However, if the _KEYNAME_ value refers to an event name, then the actual name may be of mixed case. The default for _KEYNAME_ is no keyname, designated by “.”.

STARTDATE	Contains either the date timing value or the first date timing value to use in a do-list. The default for _STARTDATE_ is no date, designated by a missing value.
ENDDATE	Contains the last date timing value to use in a do-list. The default for _ENDDATE_ is no date, designated by a missing value.
DATEINTRVL	Contains the interval for the date do-list. The default for _DATEINTRVL_ is no interval, designated by ".".
STARTDT	Contains either the datetime timing value or the first datetime timing value to use in a do-list. The default for _STARTDT_ is no datetime, designated by a missing value.
ENDDT	Contains the last datetime timing value to use in a do-list. The default for _ENDDT_ is no datetime, designated by a missing value.
DTINTRVL	Contains the interval for the datetime do-list. The default for _DTINTRVL_ is no interval, designated by ".".
STARTOBS	Contains either the observation number timing value or the first observation number timing value to use in a do-list. The default for _STARTOBS_ is no observation number, designated by a missing value.
ENDOBS	Contains the last observation number timing value to use in a do-list. The default for _ENDOBS_ is no observation number, designated by a missing value.
OBSINTRVL	Contains the interval length of the observation number do-list. The default for _OBSINTRVL_ is no interval, designated by ".".
TYPE	Type of EVENT. The default for _TYPE_ is POINT.
VALUE	Value for nonzero observation. The default for _VALUE_ is 1.0.
PULSE	INTERVAL that defines the units for the DURATION values. The default for _PULSE_ is no interval, designated by ".".
_DUR_BEFORE_	Number of durations before the timing value. The default for _DUR_BEFORE_ is 0.
_DUR_AFTER_	Number of durations after the timing value. The default for _DUR_AFTER_ is 0.
_SLOPE_BEFORE_	For TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, this determines whether the curve is GROWTH or DECAY before the timing value. The default for _SLOPE_BEFORE_ is GROWTH.
_SLOPE_AFTER_	For TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, this determines whether the curve is GROWTH or DECAY after the timing value. The default for _SLOPE_AFTER_ is GROWTH unless TYPE=TC, then the default is DECAY.
SHIFT	Number of PULSE=intervals to shift the timing value. The shift may be positive (forward in time) or negative (backward in time). If PULSE= is not specified, then the shift is in observations. The default for _SHIFT_ is 0.

TCPARM	Parameter for EVENT of TYPE=TC. The default for _TCPARM_ is 0.5.
RULE	Rule to use when combining events or when timing values of an event overlap. The default for _RULE_ is ADD.
PERIOD	Frequency interval at which the event should be repeated. If this value is missing, then the event is not periodic. The default for _PERIOD_ is no interval, designated by ".".
LABEL	Label or description for the event. If the user does not specify a label, then the default label value will be displayed as ".". For events which produce dummy variables, either the user-supplied label or the default label will be used. For COMPLEX events, the _LABEL_ value is merely a description of the group of events. See the LABEL= option for more information on the default label.

Printed Output

The HPFEVENTS procedure has no printed output other than warning and error messages as recorded in the log.

Examples

Example 12.1. The Use of Multiple Timing Values in a Single Event vs. Using Multiple Events and the EVENTCOMB Statement

This example illustrates how the HPFEVENTS procedure interprets multiple timing values that overlap and the results of the same timing values used in separate EVENTDEF statements that are combined using EVENTCOMB. Airline sales data are used for this illustration.

```
proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month start='01Jan1949'D end='01Feb1950'D;
   eventdef xmasrp= CHRISTMAS / BEFORE=(SLOPE=GROWTH DURATION=3)
                                 TYPE=RAMP RULE=MIN ;
   eventdef easterrp= EASTER / BEFORE=(SLOPE=GROWTH DURATION=3)
                                TYPE=RAMP RULE=MIN ;
   eventdef twotiming= EASTER CHRISTMAS /
                                BEFORE=(SLOPE=GROWTH DURATION=3)
                                TYPE=RAMP RULE=MIN ;
   eventcomb twoevents= easterrp xmasrp / RULE=MIN ;
   eventdata  out= evdsout1 (label='EASTER and CHRISTMAS Ramps');
   eventdummy  out= evdumout1 (label='Combining Timing Values');
run;
```

```
proc print data=evdumout1;
run;
```

Output 12.1.1. Multiple Timing Values vs. Multiple Events

```
   Obs       DATE     AIR     xmasrp     easterrp     twotiming     twoevents

     1     JAN1949    112    0.00000     0.00000      0.00000       0.00000
     2     FEB1949    118    0.00000     0.33333      0.33333       0.00000
     3     MAR1949    132    0.00000     0.66667      0.66667       0.00000
     4     APR1949    129    0.00000     1.00000      1.00000       0.00000
     5     MAY1949    121    0.00000     1.00000      1.00000       0.00000
     6     JUN1949    135    0.00000     1.00000      1.00000       0.00000
     7     JUL1949    148    0.00000     1.00000      1.00000       0.00000
     8     AUG1949    148    0.00000     1.00000      1.00000       0.00000
     9     SEP1949    136    0.00000     1.00000      0.00000       0.00000
    10     OCT1949    119    0.33333     1.00000      0.33333       0.33333
    11     NOV1949    104    0.66667     1.00000      0.66667       0.66667
    12     DEC1949    118    1.00000     1.00000      1.00000       1.00000
    13     JAN1950    115    1.00000     0.00000      0.00000       0.00000
    14     FEB1950    126    1.00000     0.33333      0.33333       0.33333
```

In this example, the ramp for Christmas is defined for observations 9 through 14. When XMASRP is evaluated, the undefined values in observations 1 through 8 are replaced with zeroes. The ramp for Easter is defined for the entire time series, as shown in the variable EASTERRP. When both timing values are used in one EVENTDEF statement for variable TWOTIMING, the values from the Easter ramp are used in observations 1 through 8, and the RULE=MIN is applied to observations 9 through 14. For the EVENTCOMB statement that defines the variable TWOEVENTS, the RULE=MIN option applies to all observations in the series.

Example 12.2. Using a DATA Step to Construct an Events Data Set

The following example uses the DATA step to automatically construct potential outliers related to the price data found in the data set SASHELP.PRICEDATA.

```
data orders(keep=date region line product sale);
     set SASHELP.PRICEDATA;
     format date monyy.;
run;
```

The following SAS code constructs an EVENTDATA IN= data set for potential outliers (identified as $sale > 450$). Only the _NAME_ and _STARTDATE_ variable are needed.

```
data outliers(keep=_NAME_ _STARTDATE_  );
     set orders;
     if (sale > 450) then do;
        _NAME_ = trim('AO')||trim(left(put(YEAR(date),8.)))||'_'
                  ||trim(left(put(MONTH(date),8.)));
        _STARTDATE_=date;
        end;
     else delete;
     format _STARTDATE_ monyy.;
run;
```

Next, identify which outliers apply to each product.

```
data product_event_list (keep= region line product _NAME_);
     set orders;
     if (sale > 450) then do;
        _NAME_ = trim('AO')||trim(left(put(YEAR(date),8.)))||'_'
                 ||trim(left(put(MONTH(date),8.)));
        end;
     else delete;
run;
```

The potential outliers in the data set OUTL_REG1_LINE1_PROD1 apply to Region 1, Line 1, and Product 1.

```
data outl_reg1_line1_prod1;
     set product_event_list;
     if ((region ~= 1) | (line ~= 1) | (product ~= 1)) then delete;
run;
```

Dummy variables are created and duplicate outlier events are eliminated from the events definition data set.

```
proc hpfevents data=orders ;
   id date interval=month;
   by region line product;
   eventdata  in= outliers ;
   eventdata  out= outldatabase (label='outlier definitions') condense;
   eventdummy out= dummies (label='dummy variables');
run;


proc print data=outldatabase;
run;


proc print data=outl_reg1_line1_prod1;
run;
```

Examining the data set OUTL_REG1_LINE1_PROD1 shows that we might want to look for outliers for May 1998, October 1999, March 2000, February 2001, June 2001, and September 2002.

Output 12.2.1. Potential Outliers for Region 1, Line 1, Product 1

```
Obs    region      line    product     _NAME_

 1        1          1         1      AO1998_5
 2        1          1         1      AO1999_10
 3        1          1         1      AO2000_3
 4        1          1         1      AO2001_2
 5        1          1         1      AO2001_6
 6        1          1         1      AO2002_9
```

PROC HPFEVENTS produced this data set, which is condensed and has the duplicate events eliminated.

Output 12.2.2. Event Definition Data Set

```
                    Obs     _NAME_       _STARTDATE_

                      1    AO1999_1       01JAN1999
                      2    AO1999_11      01NOV1999
                      3    AO2000_12      01DEC2000
                      4    AO2002_1       01JAN2002
                      5    AO1998_10      01OCT1998
                      6    AO1999_8       01AUG1999
                      7    AO2000_4       01APR2000
                      8    AO2001_1       01JAN2001
                      9    AO2001_12      01DEC2001
                     10    AO2002_12      01DEC2002
                     11    AO2000_11      01NOV2000
                     12    AO2001_3       01MAR2001
                     13    AO2001_11      01NOV2001
                     14    AO2002_2       01FEB2002
                     15    AO2000_10      01OCT2000
                     16    AO2001_2       01FEB2001
                     17    AO2001_4       01APR2001
                     18    AO2002_3       01MAR2002
                     19    AO1998_11      01NOV1998
                     20    AO1999_9       01SEP1999
                     21    AO2001_7       01JUL2001
                     22    AO1998_1       01JAN1998
                     23    AO1998_2       01FEB1998
                     24    AO1998_3       01MAR1998
                     25    AO1998_4       01APR1998
                     26    AO1998_5       01MAY1998
                     27    AO1998_6       01JUN1998
                     28    AO1998_7       01JUL1998
                     29    AO1998_8       01AUG1998
                     30    AO1998_12      01DEC1998
                     31    AO1999_2       01FEB1999
                     32    AO1999_3       01MAR1999
                     33    AO1999_4       01APR1999
                     34    AO1999_5       01MAY1999
                     35    AO1999_6       01JUN1999
                     36    AO1999_7       01JUL1999
                     37    AO1999_12      01DEC1999
                     38    AO2000_1       01JAN2000
                     39    AO2000_2       01FEB2000
                     40    AO2000_3       01MAR2000
                     41    AO2000_5       01MAY2000
                     42    AO2000_6       01JUN2000
                     43    AO2000_7       01JUL2000
                     44    AO2000_8       01AUG2000
                     45    AO2000_9       01SEP2000
                     46    AO2001_5       01MAY2001
                     47    AO2001_6       01JUN2001
                     48    AO2001_8       01AUG2001
                     49    AO2001_10      01OCT2001
                     50    AO2002_4       01APR2002
                     51    AO2002_5       01MAY2002
                     52    AO2002_7       01JUL2002
                     53    AO2002_8       01AUG2002
                     54    AO2002_9       01SEP2002
                     55    AO2002_10      01OCT2002
                     56    AO2002_11      01NOV2002
                     57    AO1998_9       01SEP1998
                     58    AO1999_10      01OCT1999
                     59    AO2001_9       01SEP2001
                     60    AO2002_6       01JUN2002
```

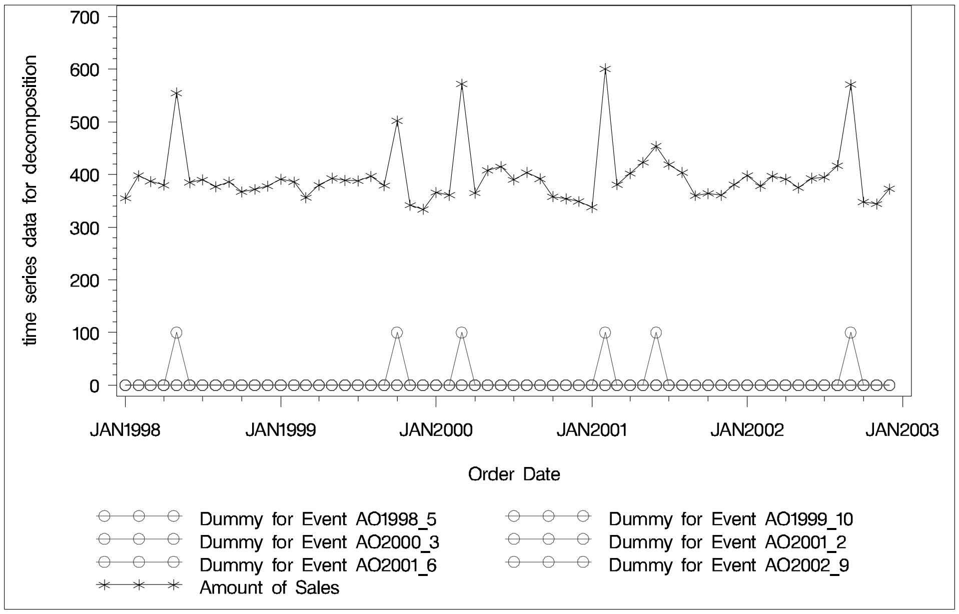
Select the observations related to Region 1, Line 1, Product 1 and scale the dummies that apply so that they are visible when plotted with the original data.

```
data pid1;
     set dummies;
     if ((region ~= 1) | (line ~= 1) | (product ~= 1)) then delete;
     else do;
         AO1998_5  = 100 * AO1998_5;
         AO1999_10 = 100 * AO1999_10;
         AO2000_3  = 100 * AO2000_3;
         AO2001_2  = 100 * AO2001_2;
         AO2001_6  = 100 * AO2001_6;
         AO2002_9  = 100 * AO2002_9;
     end;
run;
```

Use PROC GPLOT to visually verify that these potential outliers are appropriate for the original data.

```
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=('Dummy for Event AO1998_5'
                          'Dummy for Event AO1999_10'
                          'Dummy for Event AO2000_3'
                          'Dummy for Event AO2001_2'
                          'Dummy for Event AO2001_6'
                          'Dummy for Event AO2002_9'
                          'Amount of Sales');

proc gplot data=pid1 ;
      plot AO1998_5    * date = 2
           AO1999_10   * date = 2
           AO2000_3    * date = 2
           AO2001_2    * date = 2
           AO2001_6    * date = 2
           AO2002_9    * date = 2
           sale        * date = 1 / overlay legend=legend1 vaxis=axis2;
run;
```

Output 12.2.3. Plot of Amount and Dummy



Example 12.3. Preparing a Data Set for PROC HPFENGINE

This example illustrates how the HPFEVENTS procedure can be used to include events in the automatic forecasting of time series data. The data has been altered by adding a levelshift of 100 beginning at October, 1980. PROC HPFEVENTS is used to create an event named PROMOTION as a level shift occurring at October 1, 1980. PROC HPFENGINE identifies the parameter of the event PROMOTION as 97.6728, which is used in conjunction with the model named SP1 described as “ARIMA$(0, 1, 1)$ No Intercept”.

```
* make data set;
data work_intv;
     set sashelp.workers;
     if date >= '01oct80'd then electric = electric+100;
     drop masonry;
run;


* define event 'promotion';
proc hpfevents data=work_intv lead=12;
   id date interval=month;
   eventdef promotion=  '01oct80'd / TYPE=LS;
   eventdata out= evdsout1 (label='list of events');
run;
```

```
* make specs;
proc hpfarimaspec modelrepository=sasuser.mycat
              specname=sp1
              speclabel="ARIMA(0,1,1) No Intercept";
   dependent symbol=Y q=1 diflist=1 noint;
run;

proc hpfarimaspec modelrepository=sasuser.mycat
              specname=sp2
              speclabel="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
   dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
run;
```

```
* make selection list;
proc hpfselect modelrepository=sasuser.mycat
               selectname=myselect
               selectlabel="My Selection List";
   select select=mape holdout=12;
   spec sp1 sp2/
        inputmap(symbol=Y var=electric)
        eventmap(symbol=_NONE_ event=promotion)
        ;
run;


proc hpfengine data=work_intv lead=12 outest=outest
     globalselection=myselect
     modelrepository=sasuser.mycat
     inevent=evdsout1;
   id date interval=month;
   forecast electric / task = select;
run;


proc print data=outest; run;
```

Output 12.3.1. Model Selection Using Events

```
                                                               _     _
                                                               T     C
                        _                               _      R     O
                        M                               V      A     M
             _          O                               A      N     P
             S     _    D                  _            R      S     O
    _        E     M    E                  D            T      F     N
    N        L     O    L                  S            Y      O     E
O   A        E     D    V                  V            P      R     N
b   M        C     E    A                  A            E      M     T
s   E        T     L    R                  R
    _        _     _    _                  _            _      _     _

1  ELECTRIC  MYSELECT  SP1  Y          ELECTRIC   DEPENDENT  NONE  MA
2  ELECTRIC  MYSELECT  SP1  PROMOTION  PROMOTION  EVENT      NONE  SCALE



   _                                           _         _                  _   _
   F        _           _   _                  S         T                  P   S
   A        S           L   S                  T         V                  V   T
   C    _   H     _     A   C        _         D         A                  A   A
   T    L   I     P     B   O        E         E         L                  L   T
O  O    A   F     A     E   R        S         R         U                  U   U
b  R    G   T     R     L   E        T         R         E                  E   S
s  _    _   _     M     _   _        _         _         _                  _   _

1  1    1   0   MA1_1            -0.5260   0.10798    -4.8718   .000007605   0
2  0    0   0   SCALE            97.6728   4.28766    22.7800   2.0879E-32   0
```

Example 12.4. Using SAS Predefined Event Keywords Directly in Other SAS Procedures

In Example 12.3, the user can modify the SAS code to use the EVENTS system directly without using PROC HPFEVENTS. Instead of creating an event named PROMOTION, the user can use the SAS predefined event keyword LS01OCT1980D. This example uses the data set from Example 12.3 and the EVENT statement in PROC HPFDIAGNOSE to illustrate this method.

```
 proc hpfdiag data=work_intv print=all seasonality=12;
   id date interval=month;
   forecast electric;
   event LS01OCT1980D;
   trend diff=1 sdiff=1;
   arimax ;
run;
```

The output from PROC HPFDIAGNOSE shows that a model was selected that included the level shift occurring at October 1980.

Output 12.4.1. Using the EVENT Statement in PROC HPFDIAGNOSE

```
                            The HPFDIAGNOSE Procedure

                          Minimum Information Criterion

         Lags      MA 0      MA 1      MA 2      MA 3      MA 4      MA 5

         AR 0  5.883729  5.949528   6.02335  6.096772  6.170614  6.241918
         AR 1  5.949355  6.023199  6.096221  6.169567  6.243324  6.314554
         AR 2  6.023143  6.096346  6.170056   6.24107  6.314917   6.38698
         AR 3  6.096554  6.169837  6.240963  6.314829  6.387728  6.459893
         AR 4  6.170396  6.243647  6.314666  6.387965  6.461478  6.533759
         AR 5  6.241435  6.314684  6.386682  6.459847  6.533716   6.60647


                                   Functional
                               Transformation Test

                                       Functional
                              Variable    Transform

                              ELECTRIC    NONE


                            ARIMA Model Specification

           Functional                                        Model
 Variable Transform  Constant  p  d  q  P  D  Q Seasonality Criterion Statistic

 ELECTRIC NONE       NO        0  1  0  0  1  1          12 RMSE        13.6804
                            ARIMA Model Specification

                            Variable Status

                            ELECTRIC OK


                              ARIMA Event Selection

            Event Name      Selected      d     D    Status

            LS01OCT1980D    YES           1     1    OK


                ARIMA Model Specification After Adjusting for Events

             Functional                                                Model
   Variable Transform  Constant  p  d  q  P  D  Q Seasonality Event Criterion

   ELECTRIC NONE       NO        0  1  0  0  1  1          12     1 RMSE

                            ARIMA Model Specification
                           After Adjusting for Events

                     Variable Statistic    Status

                     ELECTRIC    3.3460    OK
```

Example 12.5. Viewing Dummy Variables Using SASGRAPH

This example illustrates how the HPFEVENTS procedure can be used to create dummies. These dummy variables can then be viewed using SASGRAPH. This example also shows the behavior of ramp variables when used with the SLOPE= option.

```
proc hpfevents data=sashelp.air ;
   var air;
   id date interval=month start='01Jan1951'D end='31Dec1951'D;
   eventdef infgg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=ALL)
                                    AFTER=(SLOPE=GROWTH DURATION=ALL);
   eventdef infgd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=ALL)
                                    AFTER=(SLOPE=DECAY DURATION=ALL) ;
   eventdef infdg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=ALL)
                                    AFTER=(SLOPE=GROWTH DURATION=ALL)  ;
   eventdef infdd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=ALL)
                                    AFTER=(DURATION=ALL SLOPE=DECAY) ;

   eventdef minfgg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4)
                                     AFTER=(SLOPE=GROWTH DURATION=ALL) ;
   eventdef minfgd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4)
                                     AFTER=(SLOPE=DECAY DURATION=ALL) ;
   eventdef minfdg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                     AFTER=(SLOPE=GROWTH DURATION=ALL);
   eventdef minfdd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                     AFTER=(SLOPE=DECAY DURATION=ALL) ;

   eventdef monlygg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4);
   eventdef monlygd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4)
                                      AFTER=(SLOPE=DECAY);
   eventdef monlydg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                      AFTER=(SLOPE=GROWTH) ;
   eventdef monlydd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                      AFTER=(SLOPE=DECAY);

   eventdef ninfgg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=ALL)
                                     AFTER=(SLOPE=GROWTH DURATION=2) ;
   eventdef ninfgd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=ALL)
                                     AFTER=(SLOPE=DECAY DURATION=2) ;
   eventdef ninfdg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=ALL)
                                     AFTER=(SLOPE=GROWTH DURATION=2) ;
   eventdef ninfdd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=ALL)
                                     AFTER=(SLOPE=DECAY DURATION=2) ;

   eventdef nonlygg= '01Jun1951'D  /  TYPE=RAMP AFTER=(SLOPE=GROWTH DURATION=2);
   eventdef nonlygd= '01Jun1951'D  /  TYPE=RAMP AFTER=(SLOPE=DECAY DURATION=2)
                                      BEFORE=(SLOPE=GROWTH) ;
   eventdef nonlydg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY)
                                      AFTER=(SLOPE=GROWTH DURATION=2) ;
   eventdef nonlydd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY)
                                      AFTER=(SLOPE=DECAY DURATION=2) ;

   eventdef mngg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4)
                                   AFTER=(SLOPE=GROWTH DURATION=2) ;
   eventdef mngd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=GROWTH DURATION=4)
                                   AFTER=(SLOPE=DECAY DURATION=2) ;
   eventdef mndg= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                   AFTER=(SLOPE=GROWTH DURATION=2) ;
```

```
      eventdef mndd= '01Jun1951'D  /  TYPE=RAMP BEFORE=(SLOPE=DECAY DURATION=4)
                                      AFTER=(SLOPE=DECAY DURATION=2) ;

      eventdata  out= rampds (label='Ramps Using DURATION= and SLOPE=');
      eventdummy  out= rampdummies (label='Dummy Variables for Ramps');
   run;
```

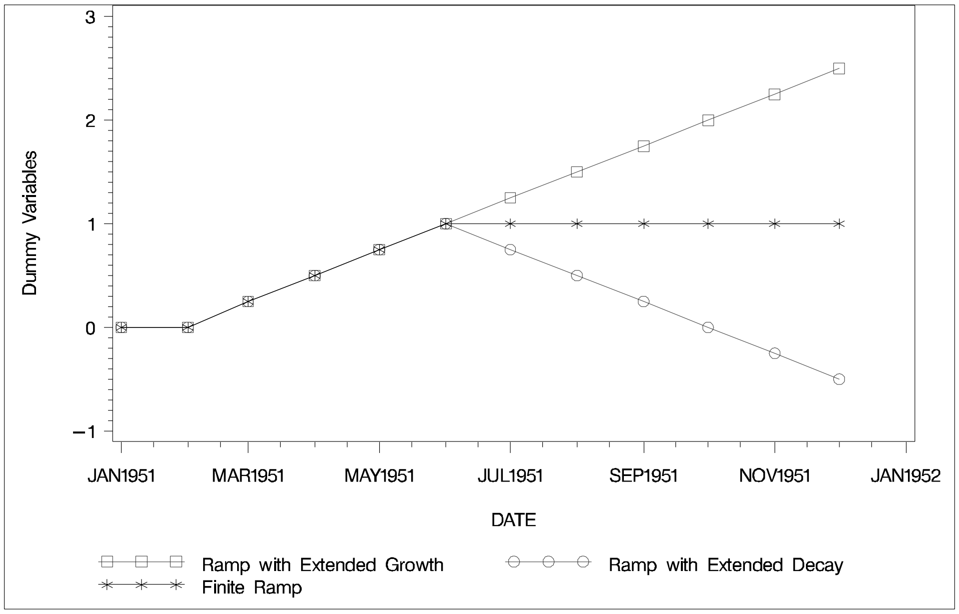
```
   proc print data=rampdummies;
   run;
```

Output 12.5.1. Ramp Dummy Variables

```
                                                  m    m    m    m
                            m    m     m    m     o    o    o    o
                i  i i  i   i    i     i    i     n    n    n    n
         D      n  n n  n   n    n     n    n     l    l    l    l
 O       A  A   f  f f  f   f    f     f    f     y    y    y    y
 b       T  I   g  g d  d   g    g     d    d     g    g    d    d
 s       E  R   g  d g  d   g    d     g    d     g    d    g    d

 1 JAN1951 145 -5 -5 5  5 0.00  0.00 1.00  1.00 0.00 0.00 1.00 1.00
 2 FEB1951 150 -4 -4 4  4 0.00  0.00 1.00  1.00 0.00 0.00 1.00 1.00
 3 MAR1951 178 -3 -3 3  3 0.25  0.25 0.75  0.75 0.25 0.25 0.75 0.75
 4 APR1951 163 -2 -2 2  2 0.50  0.50 0.50  0.50 0.50 0.50 0.50 0.50
 5 MAY1951 172 -1 -1 1  1 0.75  0.75 0.25  0.25 0.75 0.75 0.25 0.25
 6 JUN1951 178  0  0 0  0 1.00  1.00 0.00  0.00 1.00 1.00 0.00 0.00
 7 JUL1951 199  1 -1 1 -1 1.25  0.75 0.25 -0.25 1.00 1.00 0.00 0.00
 8 AUG1951 199  2 -2 2 -2 1.50  0.50 0.50 -0.50 1.00 1.00 0.00 0.00
 9 SEP1951 184  3 -3 3 -3 1.75  0.25 0.75 -0.75 1.00 1.00 0.00 0.00
10 OCT1951 162  4 -4 4 -4 2.00  0.00 1.00 -1.00 1.00 1.00 0.00 0.00
11 NOV1951 146  5 -5 5 -5 2.25 -0.25 1.25 -1.25 1.00 1.00 0.00 0.00
12 DEC1951 166  6 -6 6 -6 2.50 -0.50 1.50 -1.50 1.00 1.00 0.00 0.00

                          n    n    n    n
      n     n   n    n    o    o    o    o
      i     i   i    i    n    n    n    n
      n     n   n    n    l    l    l    l      m       m     m      m
 O    f     f   f    f    y    y    y    y      n       n     n      n
 b    g     g   d    d    g    g    d    d      g       g     d      d
 s    g     d   g    d    g    d    g    d      g       d     g      d

 1 -2.5  -1.5  2.5  3.5  0.0  1.0  0.0  1.0 0.00000  0.00  1.00  1.00000
 2 -2.0  -1.0  2.0  3.0  0.0  1.0  0.0  1.0 0.00000  0.00  1.00  1.00000
 3 -1.5  -0.5  1.5  2.5  0.0  1.0  0.0  1.0 0.16667  0.25  0.75  0.83333
 4 -1.0   0.0  1.0  2.0  0.0  1.0  0.0  1.0 0.33333  0.50  0.50  0.66667
 5 -0.5   0.5  0.5  1.5  0.0  1.0  0.0  1.0 0.50000  0.75  0.25  0.50000
 6  0.0   1.0  0.0  1.0  0.0  1.0  0.0  1.0 0.66667  1.00  0.00  0.33333
 7  0.5   0.5  0.5  0.5  0.5  0.5  0.5  0.5 0.83333  0.50  0.50  0.16667
 8  1.0   0.0  1.0  0.0  1.0  0.0  1.0  0.0 1.00000  0.00  1.00  0.00000
 9  1.0   0.0  1.0  0.0  1.0  0.0  1.0  0.0 1.00000  0.00  1.00  0.00000
10  1.0   0.0  1.0  0.0  1.0  0.0  1.0  0.0 1.00000  0.00  1.00  0.00000
11  1.0   0.0  1.0  0.0  1.0  0.0  1.0  0.0 1.00000  0.00  1.00  0.00000
12  1.0   0.0  1.0  0.0  1.0  0.0  1.0  0.0 1.00000  0.00  1.00  0.00000
```

```
axis2 label=(angle=90  'Dummy Variables');
symbol1 i=join v='star' c=black;
symbol2 i=join v='circle' c=red;
symbol3 i=join v='square' c=red;
legend1 label=none value=('Ramp with Extended Growth'
                          'Ramp with Extended Decay'
                          'Finite Ramp');

proc gplot data=rampdummies ;
      plot minfgg      * date = 3
           minfgd      * date = 2
           monlygg     * date = 1 / overlay legend=legend1 vaxis=axis2;
run;
```

Output 12.5.2. Plot of Finite and Extended Dummy Variables



References

Montes, M. (2001a), *Calculation of the Ecclesiastical Calendar*, http://www.smart.net/˜mmontes/ec-cal.html.

Montes, M. (2001b), *Nature (1876) Algorithm for Calculating the Date of Easter in the Gregorian Calendar*, http://www.smart.net/˜mmontes/nature1876.html.

Chapter 13
The HPFEXMSPEC Procedure

Chapter Contents

Chapter 13
The HPFEXMSPEC Procedure

Overview

The HPFEXMSPEC procedure creates model specifications files for external models (EXM). External model specifications are used for forecasts that are provided external to the system. These external forecasts may have originated from an external statistical model from another software package, may have been provided by an outside organization (e.g., a marketing organization, or government agency), or may be based on judgment.

External forecasts may or may not provide prediction standard errors. If the prediction standard errors are not provided, they must be computed from the prediction errors and additional information. To properly compute the prediction standard errors, the autocovariances of model residuals and information about any transformations applied to the actual time series is needed. Since the autocovariances or transformations are not known to the system, this information must be specified by the user or approximated from the actual time series or the prediction errors.

External forecasts may or may not provide lower and upper confidence limits. If lower and upper confidence limits are not provided, they must be computed from the prediction standard errors.

The external model specification is a means by which the user can specify information about how external forecasts were created so that the prediction standard errors and/or confidence limits can be approximated when they are not provided with the external forecasts.

Getting Started

The following example shows how to create an external model specification file.

```
proc hpfexmspec repository=sasuser.mymodels
                name=myexternal
                label="My External Model";
  exm method=wn;
run;
```

The options in the PROC HPFEXMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog SASUSER.MYMODELS, the NAME= option specifies that the name of the file be “myexternal.xml”, and the LABEL= option specifies a label for this catalog member. The EXM statement in the procedure specifies the external model and the options used to control the parameter estimation process for the model.

Syntax

The following statements are used with the HPFEXMSPEC procedure.

> **PROC HPFEXMSPEC** *options*;
> **EXM** *options*;

Functional Summary

The statements and options controlling the HPFEXMSPEC procedure are summarized in the following table.

Description	Statement	Option
Statements		
specifies exponential smoothing model	EXM	
Model Repository Options		
specifies the model repository	PROC HPFEXMSPEC	REPOSITORY=
specifies the model specification name	PROC HPFEXMSPEC	NAME=
specifies the model specification label	PROC HPFEXMSPEC	LABEL=
External Model Options		
specifies the time series transformation	EXM	TRANSFORM=
specifies median forecasts	EXM	MEDIAN
specifies the method of creating forecast standard errors	EXM	METHOD=
specifies the number of parameters used to create the external forecast	EXM	NPARMS=
specifies the number of time lags used to compute the autocorrelations	EXM	NLAGPCT=
specifies that the external model parameters are fixed values	EXM	NOEST

PROC HPFEXMSPEC Statement

> **PROC HPFEXMSPEC** *options ;*

The following options can be used in the PROC HPFEXMSPEC statement.

LABEL= *SAS-label*
labels the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

NAME= *SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

REPOSITORY= *SAS-catalog-name | SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

EXM Statement

EXM *options ;*

The EXM statement is used to specify an external model used to generate the external forecasts. These options are not needed if the prediction standard errors and confidence limits are provided.

The following examples illustrate typical uses of the EXM statement:

```
/* default specification */
exm;

/* Actual Series Autocorrelation */
exm method=acf;

/* Prediction Error Autocorrelation */
exm method=erroracf;
```

The following options can be specified in the EXM statement:

TRANSFORM= *option*
specifies the time series transformation that was applied to the actual time series when generating the external forecast. The following transformations are provided:

NONE	No transformation is applied.
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 actual time series must be strictly positive. Once the time series is transformed, the model parameters are estimated using the transformed time series. The forecasts of the transformed time series are then computed, and finally, the transformed time series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

MEDIAN
specifies that the median forecast values were used to generate the external forecasts. The external forecasts may have been based on the mean or median. By default the mean value is assumed. If no transformation was used by the external forecasting method, as specified by the TRANSFORM=NONE option, the mean and median prediction standard errors and confidence limits are identical.

METHOD= *method-name*
specifies the external model to be used to approximate the prediction standard errors. The default is METHOD=ACF. The following forecasting models are provided:

NONE	No prediction error autocorrelation.
WN	Prediction error autocorrelation is white noise.
ACF	Autocorrelation is used.
ERRORACF	Prediction error autocorrelation is used.
PERFECT	Perfect autocorrelation is assumed.

NPARMS=*n*
specifies the number of parameters used by the external model to generate the forecasts. The default is NPARMS=0.

NLAGPCT=*number*
specifies the number of time lags as a percentage of the number of computed predictions errors. The default is NLAGPCT=0.25.

SIGMA=*number*
specifies the prediction standard error for the external model. If the SIGMA= option is specified with the NOEST option, the prediction mean square error specified by the SIGMA= option is used. Otherwise, the prediction mean square error is computed from the prediction errors using the NPARMS= option.

NOEST
specifies that the external model parameters are fixed values. To use this option, all of the external model parameters must be explicitly specified. By default, the external model parameters are optimized.

Chapter 14
The HPFIDMSPEC Procedure

Chapter Contents

Chapter 14
The HPFIDMSPEC Procedure

Overview

The HPFIDMSPEC procedure creates model specifications files for intermittent demand models (IDM).

You can specify many types of intermittent demand models using this procedure. In particular, any model that can be analyzed using the HPF procedure can be specified.

Getting Started

The following example shows how to create an intermittent demand model specification file. In this example, a model specification for *Croston's* method is created.

```
proc hpfidmspec repository=sasuser.mymodels
                name=mycroston
                label="Croston Method" ;
  idm  interval=(method=simple)
       size=(method=simple) ;
run;
```

The options in the PROC HPFIDMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog SASUSER.MYMODELS, the NAME= option specifies that the name of the file be "mycroston.xml", and the LABEL= option specifies a label for this catalog member. The IDM statement in the procedure specifies the intermittent demand model and the options used to control the parameter estimation process for the model.

Syntax

The following statements are used with the HPFIDMSPEC procedure.

PROC HPFIDMSPEC *options*;
    IDM *options*;

Functional Summary

The statements and options controlling the HPFIDMSPEC procedure are summarized in the following table.

Description	Statement	Option
Statements		
specifies the intermittent demand model	IDM	
Model Repository Options		
specifies the model repository	PROC HPFIDMSPEC	REPOSITORY=
specifies the model specification name	PROC HPFIDMSPEC	NAME=
specifies the model specification label	PROC HPFIDMSPEC	LABEL=
Intermittent Demand Model Options		
specifies the model for average demand	IDM	AVERAGE=
specifies the base value	IDM	BASE=
specifies the model for demand intervals	IDM	INTERVAL=
specifies the model for demand sizes	IDM	SIZE=

PROC HPFIDMSPEC Statement

PROC HPFIDMSPEC *options ;*

The following options can be used in the PROC HPFIDMSPEC statement.

LABEL= *SAS-label*
specified a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

NAME= *SAS-name*
names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

REPOSITORY= *SAS-catalog-name* | *SAS-file-reference*
names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

IDM Statement

IDM *options ;*

The IDM statement is used to specify an intermittent demand model. An intermittent demand series can be analyzed in two ways: individually modeling both demand interval and size component, or jointly modeling these components using the average demand component (demand size divided by demand interval). The IDM statement specifies the two smoothing models to be used to forecast the demand interval component (INTERVAL= option) and the demand size component (SIZE= option), or specifies the single smoothing model to be used to forecast the average demand component (AVERAGE= option). Therefore, two smoothing models (INTERVAL= and SIZE= options) must be specified or one smoothing model (AVERAGE= option) must be specified. Only one statement can be specified.

The following examples illustrate typical uses of the IDM statement:

```
/* default specification */
idm;

/* demand interval model only specification */
idm interval=(transform=log);

/* demand size model only specification */
idm size=(method=linear);

/* Croston's Method */
idm interval=(method=simple)
    size    =(method=simple);

/* Log Croston's Method */
idm interval=(method=simple transform=log)
    size    =(method=simple transform=log);

/* average demand model specification */
idm average=(method=bestn);
```

The default specification uses both the INTERVAL= option and SIZE= option defaults for the decomposed (Croston's) demand model and the AVERAGE= option defaults for the average demand model.

The following example illustrates how to automatically choose the decomposed demand model using MAPE as the model selection criterion:

```
idm interval=(method=simple transform=auto select=mape)
    size    =(method=simple transform=auto select=mape);
forecast sales / model=idm select=mape;
```

The preceding fits two forecast models (simple and log simple exponential smoothing) to both the demand interval and size components. The forecast model that results in the lowest in-sample MAPE for each component is used to forecast the component.

The following example illustrates how to automatically choose the average demand model using MAPE as the model selection criterion:

```
idm average=(method=simple transform=auto select=mape);
forecast sales / model=idm;
```

The preceding fits two forecast models (simple and log simple exponential smoothing) to the average demand component. The forecast model that results in the lowest in-sample MAPE is used to forecast the component.

Combining the above two examples, the following example illustrates how to automatically choose between the decomposed demand model and the average demand model using MAPE as the model selection criterion:

```
idm interval=(method=simple transform=auto select=mape)
    size    =(method=simple transform=auto select=mape)
    average =(method=simple transform=auto select=mape);
forecast sales / model=idm select=mape;
```

The preceding automatically selects between the decomposed demand model and the average demand model as described previously. The forecast model that results in the lowest in-sample MAPE is used to forecast the series.

The following options can be specified in the IDM statement:

INTERVAL=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand interval component. See the following smoothing model specification options.

SIZE=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand size component. See the following smoothing model specification options.

AVERAGE=(*smoothing-model-options*)
specifies the smoothing model used to forecast the demand average component. See the following smoothing model specification options.

BASE=*AUTO | number*
specifies the base value of the time series used to determine the demand series components. The demand series components are determined based on the departures from this base value. If a base value is specified, this value is used to determine the demand series components. If BASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's Method use BASE=0, which defines departures based on zero. The default is BASE=0.

Given a time series, y_t, and base value, b, the time series is adjusted by the base value to create the base adjusted time series, $x_t = y_t - b$. Demands are assumed to occur when the base adjusted series is nonzero (or when the time series, y_t, departs from the base value, b).

When BASE=AUTO, the base value is automatically determined by the time series median, minimum, and maximum value and the INTERMITTENT= option value of the FORECAST statement.

Smoothing Model Specification Options for IDM Statement

The smoothing model options describe how to forecast the demand interval, size, and average demand components (INTERVAL= option, SIZE= option, and AVERAGE= option).

If the smoothing model options are not specified, the following are the defaults for the demand interval, size, and average components.

```
interval=(transform=auto method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999) select=rmse bounds=(1,.));

size    =(transform=auto method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999) select=rmse);

average  =(transform=auto method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999) select=rmse);
```

The above smoothing model options provide the typical automation in intermittent demand model selection.

The following describes the smoothing model options:

TRANSFORM= *option*
specifies the time series transformation to be applied to the demand component. The following transformations are provided:

NONE	No transformation is applied.
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
AUTO	Automatically choose between NONE and LOG based on model selection criteria. This option is the default.

When the TRANSFORM= option is specified, the demand component must be strictly positive. Once the demand component is transformed, the model parameters are estimated using the transformed component. The forecasts of the transformed component are then computed, and finally, the transformed component forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified.

MEDIAN
specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default the mean value is provided. If no transformation is applied to the actual series using the TRANSFORM= option, the mean and median component forecast values are identical.

METHOD= *method-name*
specifies the forecasting model to be used to forecast the demand component. A single model can be specified or a group of candidate models can be specified. If a group of models is specified, the model used to forecast the accumulated time series is selected based on the SELECT= option of the IDM statement and the HOLDOUT= option of the FORECAST statement. The default is METHOD=BESTN. The following forecasting models are provided:

SIMPLE	Simple (Single) Exponential Smoothing
DOUBLE	Double (Brown) Exponential Smoothing
LINEAR	Linear (Holt) Exponential Smoothing
DAMPTREND	Damped Trend Exponential Smoothing
BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)

NOSTABLE
specifies that the smoothing model parameters are not restricted to the additive invertible region of the parameter space. By default, the smoothing model parameters are restricted to be inside the additive invertible region of the parameter space.

LEVELPARM= *number*
specifies the level weight parameter initial value. See the following smoothing model parameter specifications.

LEVELREST=(*number*,*number***)**
specifies the level weight parameter restrictions. See the following smoothing model parameter specifications.

TRENDPARM= *number*
specifies the trend weight parameter initial value. See the following smoothing model parameter specifications.

TRENDREST=(*number*,*number***)**
specifies the trend weight parameter restrictions. See the following smoothing model parameter specifications.

DAMPPARM= *number*
specifies the damping weight parameter initial value. See the following smoothing model parameter specifications.

DAMPREST=(*number*,*number***)**
specifies the damping weight parameter restrictions. See the following smoothing model parameter specifications.

NOEST

specifies that the smoothing model parameters are fixed values. To use this option, all of the smoothing model parameters must be explicitly specified. By default, the smoothing model parameters are optimized.

BOUNDS=(*number*,*number*)

Specifies the component forecast bound. See the following smoothing model forecast bounds.

SELECT= *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option would often be used in conjunction with the HOLDOUT= option specified in the FORECAST statement. The default is SELECT=RMSE. The statistics of fit provided are the same as those provided in the FORECAST statement.

Smoothing Model Parameter Specification Options

The parameter options are used to specify smoothing model parameters. If the parameter restrictions are not specified the default is (0.001 0.999), which implies that the parameters are restricted between 0.001 and 0.999. Parameters and their restrictions are required to be greater than or equal to -1 and less than or equal to 2. Missing values indicate no lower and/or upper restriction. If the parameter initial values are not specified, the optimizer uses a grid search to find an appropriate initial value.

Smoothing Model Forecast Bounds Options

Specifies the demand component forecast bounds. The forecast bounds restrict the component forecasts. The default for demand interval forecasts is BOUNDS=1. The lower bound for the demand interval forecast must be greater than one. The default for demand size forecasts is BOUNDS=(.,.) or no bounds. The demand size forecasts bounds are applied after the forecast is adjusted by the base value.

Chapter 15
The HPFRECONCILE Procedure

Chapter Contents

Chapter 15
The HPFRECONCILE Procedure

Overview

When the data are organized in a hierarchical fashion, there are often accounting constraints that link series at different levels of the hierarchy. For example, the sales of a particular product by a retail company is the sum of the sales of the same product in all stores belonging to the company. It seems natural to require that the same constraints hold for the predicted values as well. However, imposing such constraints during the forecasting process can be difficult or impossible. Therefore, the series are often forecast independently at different levels so that the resulting forecasts do not abide by the constraints binding the original series. The after-the-fact process through which such constraints are enforced is called *reconciliation*.

The HPFRECONCILE procedure reconciles forecasts of time series data at two different levels of aggregation. Optionally, the HPFRECONCILE procedure can disaggregate forecasts from upper-level forecasts or aggregate forecasts from lower-level forecasts.

Additionally, the procedure enables the user to specify the direction and the method of reconciliation, equality constraints, and bounds on the reconciled values at each point in time.

Getting Started

This section outlines the use of the HPFRECONCILE procedure.

Consider the following hierarchical structure of the SASHELP.PRICEDATA data set.

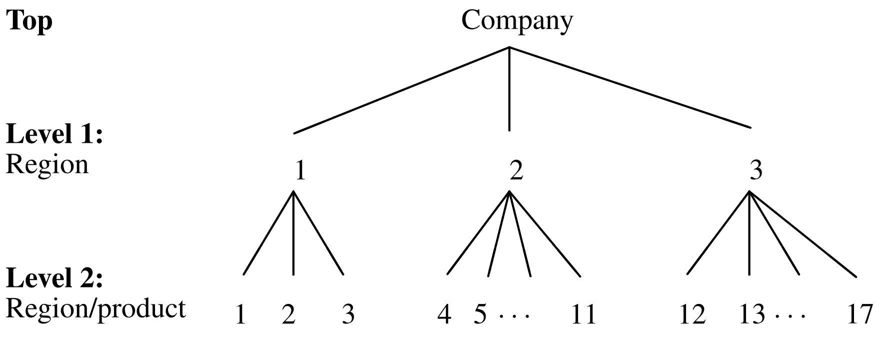


Figure 15.1. Hierarchical Structure of SASHELP.PRICEDATA

Forecasts for the dependent variable sale are generated first at level 2, region / product, and then at level 1, region. The separate forecasts are then reconciled in a bottom-up manner by using the HPFRECONCILE procedure.

```
/*
/ Forecast series at level 2 (region/product);
/-------------------------------------------------------------*/

*Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
   outest=lvl2est
   modelrepository=work.mycat
   prefilter=both
   criterion=mape;
   id date interval=month;
   by region product;
   forecast sale;
   input price;
run;

*Step 2: estimation and forecasting ;
proc hpfengine data=sashelp.pricedata inest=lvl2est
   out=_null_ outest=lvl2fest
   modelrepository=work.mycat outfor=lvl2for;
   id date interval=month;
   by region product;
   forecast sale / task=select ;
   stochastic price;
run;

/*
/ Forecast aggregated series at level 1 (region);
/-------------------------------------------------------------*/
*Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
   outest=lvl1est
   modelrepository=work.mycat
   prefilter=both
   criterion=mape;
   id date interval=month notsorted;
   by region;
   forecast sale / accumulate=total;
   input price / accumulate=average;
run;

*Step 2: estimation and forecasting;
proc hpfengine data=sashelp.pricedata inest=lvl1est
   out=_null_ outest=lvl1fest
   modelrepository=work.mycat outfor=lvl1for;
   id date interval=month notsorted;
   by region;
   forecast sale / task=select accumulate=total;
   stochastic price /accumulate=average;
run;


/*
/ Reconcile forecasts bottom up with default settings
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2for aggdata=lvl1for
     direction=BU
     outfor=lvl1recfor;
   id date interval=month;
```

```
      by region product;
   run;
```

Syntax

The HPFRECONCILE procedure is controlled by the following statements:

> **PROC HPFRECONCILE** < *options* > **;**
>   **BY** *variables* < / *option* > **;**
>   **AGGBY** *variables* < / *option* > **;**
>   **ID** *variable INTERVAL=interval* < / *options* > **;**
>   **DISAGGDATA** < *options* > **;**
>   **AGGDATA** < *options* > **;**

Functional Summary

The statements and options used with the HPFRECONCILE procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify the AGGBY variables	AGGBY	
specify the BY variables	BY	
specify the time ID variable	ID	
specify custom variable names for the DISAGGDATA= data set	DISAGGDATA	
specify custom variable names for the AGGDATA= data set	AGGDATA	
Data Set Options		
specify the disaggregated input data set (child level in the hierarchy)	HPFRECONCILE	DISAGGDATA=
specify the aggregated input data set (parent level in the hierarchy)	HPFRECONCILE	AGGDATA=
specify the output data set	HPFRECONCILE	OUTFOR=
specify the data set that contains the constraints on the reconciled forecasts	HPFRECONCILE	CONSTRAINT=
specify that the OUTFOR= data sets contain the RECDIFF variable	HPFRECONCILE	RECDIFF
specify the name of the variable that contains the actual values in the DISAGGDATA= data set	DISAGGDATA	ACTUAL=
specify the name of the variable that contains the actual values in the AGGDATA= data set	AGGDATA	PREDICT=

Description	**Statement**	**Option**
specify the name of the variable that contains the predicted values in the DISAGGDATA= data set	DISAGGDATA	ACTUAL=
specify the name of the variable that contains the predicted values in the AGGDATA= data set	AGGDATA	PREDICT=
specify the name of the variable that contains the lower confidence limit in the DISAGGDATA= data set	DISAGGDATA	LOWER=
specify the name of the variable that contains the lower confidence limit in the AGGDATA= data set	AGGDATA	LOWER=
specify the name of the variable that contains the upper confidence limit in the DISAGGDATA= data set	DISAGGDATA	UPPER=
specify the name of the variable that contains the upper confidence limit in the AGGDATA= data set	AGGDATA	UPPER=
specify the name of the variable that contains the prediction error in the DISAGGDATA= data set	DISAGGDATA	ERROR=
specify the name of the variable that contains the prediction error in the AGGDATA= data set	AGGDATA	ERROR=
specify the name of the variable that contains the standard error in the DISAGGDATA= data set	DISAGGDATA	STD=
specify the name of the variable that contains the standard error in the AGGDATA= data set	AGGDATA	STD=
Error Message Options		
specify the resolution of error and warning messages	HPFRECONCILE	ERRORTRACE=
Analysis Options		
specify the aggregation method	HPFRECONCILE	AGGREGATE=
specify the confidence level	HPFRECONCILE	ALPHA=
specify the method of computing confidence limits	HPFRECONCILE	CLMETHOD=
specify the reconciliation direction	HPFRECONCILE	DIRECTION=
specify the ending time ID value	ID	END=
specify the frequency	ID	INTERVAL=
specify the disaggregation function	HPFRECONCILE	DISAGGREGATION=
specify that only the prediction is to be reconciled	HPFRECONCILE	PREDICTONLY

Description	Statement	Option
specify the method of computing standard errors	HPFRECONCILE	STDMETHOD=
specify boundaries for the standard error	HPFRECONCILE	STDDIFBD=
specify the starting time ID value	ID	START=
specify that the loss function be weighted by the inverse of the prediction variances	HPFRECONCILE	WEIGHTED

PROC HPFRECONCILE Statement

PROC HPFRECONCILE *options* **;**

The following options can be used in the PROC HPFRECONCILE statement.

Options related to the input data sets

DISAGGDATA | DATA= *SAS-data-set*

specifies the name of the SAS data set containing the forecast of the disaggregated time series data. Typically, the DISAGGDATA= data set is generated by the OUTFOR= statement of the HPFENGINE procedure.

If the DISAGGDATA= data set is not specified, the data set opened last is used. The dimensions of the DISAGGDATA= data set are greater than the dimensions of the AGGDATA= data set. The DISAGGDATA= data set must be sorted by the BY variables and by the ID variable when the latter is specified.

AGGDATA= *SAS-data-set*

specifies the name of the SAS data set containing the forecasts of the aggregated time series data. Typically, the AGGDATA= data set is generated by the OUTFOR= statement of the HPFENGINE procedure. If the AGGDATA= data set is not specified, only bottom-up reconciliation is allowed.

The AGGDATA data set must contain a proper subset, possibly empty, of the BY variables present in the DISAGGDATA data set. Such BY variables are called AGGBY variables. The AGGDATA= data sets must be sorted by the AGGBY variables and by the ID variable when the latter is specified.

CONSTRAINT= *SAS-data-set*

specifies the name of the SAS data set containing the constraints for the reconciled series. See “CONSTRAINT= Data Set” for more details.

Options Related to the Output Data Sets

OUTFOR= *SAS-data-set*

specifies the name of the output SAS data set that will contain the reconciled values.

OUTRECFAIL= *SAS-data-set*

specifies the name of the SAS data set containing a summary of the nodes for which reconciliation failed.

RECDIFF

If the RECDIFF option is specified, the OUTFOR= data sets will contain a variable named RECDIFF that is the difference between the reconciled forecasts and the original forecasts.

Options Related to Error Messages

ERRORTRACE= *option*

specifies the resolution at which the error and warning messages should be printed to the log.

The following values are allowed:

DATA	Messages are printed only one time at the end of the procedure run.
AGGBY	Messages are printed for each AGGBY group.
ID	Messages are printed for each ID value.

The default is ERRORTRACE=DATA.

Options Related to the Analysis

AGGREGATE= TOTAL | AVERAGE

specifies whether the dependent variable in the AGGDATA= data set is the total sum or the average of the dependent variable in the DISAGGDATA= data set. The default is AGGREGATE=TOTAL.

ALPHA= *n*

specifies the level of the confidence limits when CLMETHOD=GAUSSIAN. 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.

CLMETHOD= *option*

specifies the method used to compute confidence limits for the reconciled forecasts.

The following methods are provided:

GAUSSIAN	The confidence intervals are computed assuming that the forecasts are approximately Gaussian.
SHIFT	The confidence intervals are computed by recentering the original confidence intervals around the new forecasts.

The default value is CLMETHOD=SHIFT. See "Details" for more information about the methods of computing confidence intervals.

DIRECTION= *Reconciliation-Direction*

specifies the reconciliation direction. The following reconciliation values are allowed:

BU	Bottom-up reconciliation.
TD	Top-down reconciliation.

If the AGGDATA= data set is not specified, only DIRECTION=BU is allowed.

The default value is DIRECTION=BU.

See "Details" for more information about the reconciliation directions available in PROC HPFRECONCILE.

DISAGGREGATION= DIFFERENCE | PROPORTIONAL

specifies the type of loss function for top-down reconciliation.

DISAGGREGATION=PROPORTIONAL is available only when all the forecasts at a given ID value share the same sign. See "Details" for more information about on the expressions of the loss function.

The default value is DISAGGREGATION=DIFFERENCE.

PREDICTONLY

specifies that only the predicted value is to be reconciled.

SIGN= *option*

specifies the sign constraint on the reconciled series. Valid values are as follows:

NONNEGATIVE | POSITIVE	if the output series are supposed to be nonnegative.
NONPOSITIVE | NEGATIVE	if the output series are supposed to be nonpositive.

STDMETHOD= *option*

specifies the method used to compute standard errors for the reconciled forecasts.

The following methods are provided:

UNCHANGED	Reconciled standard errors are the original standard errors.
AGG	Reconciled standard errors are proportional to the original aggregated standard errors.
DISAGG	Reconciled standard errors are proportional to the original disaggregated standard errors.

The default values are STDMETHOD=DISAGG for top-down reconciliation and STDMETHOD=AGG for bottom-up reconciliation. However, if the AGGDATA= data set is not specified for bottom-up reconciliation, then STDMETHOD=DISAGG is the default. See "Details" for more information about the methods of computing standard errors.

STDDIFBD= *n*

specifies a positive number that defines boundaries for the percentage difference between the original standard error and the reconciled standard error. If the percentage difference is greater than the values specified in the STDDIFBD= option, the reconciled standard error will be equal to the boundary value. For example, if

STDDIFBD=.3, the reconciled standard errors will be within a 30% band of the original standard errors.

The default value is STDDIFBD=.25.

Experimental **WEIGHTED**
specifies that the loss function for top-down reconciliation be weighted by the inverse of the variance of the statistical forecasts.

BY Statement

BY *variables <NOTSORTED>*;

The BY statement defines separate groups of observations for the DISAGGDATA= data set. BY variables can be either character or numeric.

All BY variables must exist in the DISAGGDATA= data set. Conversely, only a subset, or none, of the BY variables must be present in the AGGDATA= data set. The BY variables that are present in the AGGDATA= data set are called AGGBY variables. PROC HPFRECONCILE finds the AGGBY variables by comparing the variables in the BY statement with the variables in the AGGDATA= data set. The AGGBY groups must follow the same sorting order in both the DISAGGDATA= and the AGGDATA= data sets. However, some groups can be missing from either data set if the NOTSORTED option is not specified. When the NOTSORTED option is specified, all AGGBY groups must be present in both data sets and must follow the same order. See the AGGBY statement for more details.

AGGBY Statement

AGGBY *variables* ;

When DIRECTION=BU and the AGGDATA= data set is not specified, the AGGBY statement can be used to specify the BY variables in the OUTFOR= data set.

If the AGGDATA= data set is specified, the AGGBY statement is ignored.

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 frequency associated with the time series. 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. If the ID statement is specified, the ID variable must be present and must have the same frequency in both the DISAGGDATA= data set and the AGGDATA= data set.

The following options can be used with the ID statement.

IRREGULAR *option*
specifies whether to allow for irregularities in the ID variable frequency. By de-

fault, irregularities are not allowed. That is, all ID values corresponding to the INTERVAL= frequency must be present between the START= and END= values in both AGGDATA= and DISAGGDATA= data sets.

END= *option*
specifies a SAS date, datetime, or time value that represents the date at which the reconciliation should end. If the largest time ID variable value is less than the END= value, this option has no effect.

INTERVAL= *interval*
specifies the frequency of the input time series. The frequency must be the same for all input data sets. For example, if the input data sets consist of quarterly observations, then INTERVAL=QTR should be used. See the *SAS/ETS User's Guide* for the intervals that can be specified.

START= *option*
specifies a SAS date, datetime, or time value that represents the time ID value at which the reconciliation should begin. This option can be used to limit the reconciliation process only to future forecasts—that is, forecasts that are outside the historical period—and reduce the computational burden. For example, START="&sysdate"D uses the automatic macro variable SYSDATE to start the reconciliation at the current date.

DISAGGDATA Statement

DISAGGDATA < *options* > **;**

The DISAGGDATA statement enables the user to specify custom names for forecasting variables in the DISAGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options are available:

- ACTUAL=variable-name
- PREDICT=variable-name
- LOWER=variable-name
- UPPER=variable-name
- ERROR=variable-name
- STD=variable-name

AGGDATA Statement

AGGDATA < *options* > **;**

The AGGDATA statement enables the user to specify custom names for forecasting variables in the DISAGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options are available:

- ACTUAL=variable-name
- PREDICT=variable-name
- LOWER=variable-name
- UPPER=variable-name
- ERROR=variable-name
- STD=variable-name

Details

Assume a two-level hierarchical structure as depicted in Figure 15.2.

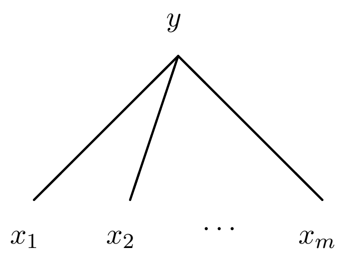


Figure 15.2. Hierarchical Structure

Let y_t be the aggregate total at time t, and let $\mathbf{x}_t = [x_{1,t}, x_{2,t}, \dots, x_{m,t}]'$ be the vector of disaggregated values at time t, $t = 1, \dots, T$. As usual, indicate by $\hat{y}_t$ and $\hat{\mathbf{x}}_t$ the pre-reconciliation statistical model forecasts of y_t and $\mathbf{x}_t$, respectively. Let $\tilde{y}_t$ and $\tilde{\mathbf{x}}_t$ indicate instead the reconciled values. The number of series m can vary with t; however, for simplicity, it is considered fixed in the following discussion.

At each time t, the values of the series $x_{i,t}$, $i = 1 \dots, m$, and y_t are bound by an aggregation constraint. By default, the constraint is assumed to be $y_t = \sum_{i=1}^{m} x_{i,t}$, which corresponds to the AGGREGATE=TOTAL option of the PROC HPFRECONCILE statement. If instead the option AGGREGATE=AVERAGE is specified, the constraint is $y_t = \frac{1}{m} \sum_{i=1}^{m} x_{i,t}$. For example, if the x_i's are the sales at store level for a retail company, then y_t can be either the total sales at company level or the average sales per store.

If you need to have forecasts at both levels of the hierarchy, it is often more convenient to produce statistical forecasts separately for each series. However, the resulting forecasts do not abide by the aggregation constraint that binds the original series. The after-the-fact process through which the statistical forecasts are modified to enforce the aggregation constraint is called *reconciliation*.

By determining whether the upper-level forecasts or the lower-level forecasts are adjusted to meet the aggregation constraint, you can distinguish between bottom-up (BU) and top-down (TD) reconciliation.

Additionally, PROC HPFRECONCILE enables you to impose constraints on the individual reconciled forecasts. For example, you can require that $\tilde{x}_1 = 10$ and $\tilde{x}_2 \geq 15$.

Top-Down Reconciliation

The goal of top-down (TD) reconciliation is to adjust the statistical forecasts $\hat{x}_{i,t}$ to obtain new series $\{\tilde{x}_{i,t}\}$ of reconciled forecasts so that the sum of the reconciled forecasts at each fixed time t is equal to $\hat{y}_t$, and satisfies the constraints that you specify in the CONSTRAINT= data set.

The problem can be restated as follows: minimize with respect to $\tilde{\mathbf{x}}$ a quadratic loss function

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t)$$

subject to the following constraints:

1. the *top-down constraint*

$$\sum_{i=1}^{m} \tilde{x}_{i,t} = \hat{y}_t$$

2. the equality constraints

$$\tilde{x}_{i,t} = e_{i,t} \qquad i \in E_t$$

3. the lower bounds

$$\tilde{x}_{i,t} \geq l_{i,t} \qquad i \in L_t$$

4. the upper bounds

$$\tilde{x}_{i,t} \leq u_{i,t} \qquad i \in U_t$$

where E_t, L_t, and U_t are subsets of $\{1, 2, \ldots, m\}$.

When needed, PROC HPFRECONCILE uses an iterative interior-point algorithm to solve the quadratic optimization problem.

When DISAGGREGATION=DIFFERENCE, the loss function is

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)' W^{-1} (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)$$

where W is a positive semidefinite matrix of weights independent of $\tilde{\mathbf{x}}_t$.

If the WEIGHTED option is specified, the matrix W is the diagonal matrix with the variances of $\hat{\mathbf{x}}_t$ on the main diagonal. Otherwise, by default, W is the identity matrix I.

Note that the loss function when DISAGGREGATION=DIFFERENCE is defined for any value of $\hat{\mathbf{x}}_t$.

When DISAGGREGATION=PROPORTIONAL, the loss function is

$$L(\tilde{\mathbf{x}}_t;\hat{\mathbf{x}}_t) = (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)'\hat{X}^{-\frac{1}{2}}W^{-1}\hat{X}^{-\frac{1}{2}}(\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)$$

where $\hat{X}^{-\frac{1}{2}}$ is a diagonal matrix with the square root of $\hat{\mathbf{x}}_t$ on the main diagonal.

Notice that when DISAGGREGATION=PROPORTIONS, the loss function is defined only when all $\hat{x}_{i,t}$ are strictly positive. However, the solutions can be extended to the cases where they are all nonnegative or they are all nonpositive. PROC HPFRECONCILE checks whether the signs of all forecasts at any given time t are concordant. If they are not, it uses DISAGGREGATION=DIFFERENCE for only those time ID values. In such a case, the _RECONSTATUS_ variable indicates for which observations the loss function used in the reconciliation process was different from the one that you specified in the PROC HPFRECONCILE statement. You can also use the ERRORTRACE=ID option to print a message to the log for each ID value for which the forecasts were not reconciled according to your specification.

The case where $\sum_{j=1}^{m} \hat{x}_{j,t} = 0$ and DISAGGREGATION=PROPORTIONS is handled by setting $\tilde{x}_{i,t} = \frac{\hat{y}_t}{m}$ when AGGREGATE=TOTAL and $\tilde{x}_{i,t} = \hat{y}_t$ when AGGREGATE=AVERAGE.

Now consider the case where the only constraint is the top-down constraint and $W = I$. Under such hypotheses, the top-down problem admits intuitive solutions.

When DISAGGREGATION=DIFFERENCE, the loss function becomes

$$L(\tilde{\mathbf{x}}_t;\hat{\mathbf{x}}_t) = \sum_{i=1}^{m}(\hat{x}_{i,t} - \tilde{x}_{i,t})^2$$

This leads to the following solution:

$$\tilde{x}_{i,t} = \hat{x}_{i,t} + \frac{r_t}{m}$$

where r_t is the aggregation error—that is,

$$r_t := \hat{y}_t - \sum_{i=1}^{m}\hat{x}_{i,t} \qquad \text{when AGGREGATE} = \text{TOTAL}$$

and

$$r_t := m\hat{y}_t - \sum_{i=1}^{m}\hat{x}_{i,t} \qquad \text{when AGGREGATE} = \text{AVERAGE}$$

Thus, when DISAGGREGATION=DIFFERENCE, the reconciled forecast $\tilde{x}_{i,t}$ is found by equally splitting the aggregation error r_t among the disaggregated forecasts $\hat{x}_{i,t}$.

Notice that even if all statistical forecasts $\hat{x}_{i,t}$ are strictly positive, the reconciled forecasts $\tilde{x}_{i,t}$ need not be so if no bounds are specified. In particular, $\hat{x}_{i,t} = 0$ does not imply $\tilde{x}_{i,t} = 0$. On the other hand, as previously mentioned, DISAGGREGATION=DIFFERENCE can be used when the statistical forecasts have discordant signs.

If DISAGGREGATION=PROPORTIONS, the loss function becomes

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = \sum_{i=1}^{m} \left(\frac{\hat{x}_{i,t} - \tilde{x}_{i,t}}{\sqrt{\hat{x}_{i,t}}} \right)^2$$

This leads to the following solutions:

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^{m} \hat{x}_{j,t}} \hat{y}_t \qquad \text{when AGGREGATE} = \text{TOTAL}$$

and

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^{m} \hat{x}_{j,t}} m\hat{y}_t \qquad \text{when AGGREGATE} = \text{AVERAGE}$$

Thus, the reconciled forecast $\tilde{x}_{i,t}$ is found by disaggregating y_t or my_t according to the proportion that $\hat{x}_{i,t}$ represents in the total sum of the disaggregated forecasts.

Missing Predicted Values

When some of the predicted values $\hat{x}_{i,t}$ are missing, the missing values are replaced by the actual values $x_{i,t}$, if these are present. This is done to prevent bias between the aggregated and reconciled forecasts, which results from models in which missing values in the predictions are generated because of the presence of lagged variables.

Standard Errors

When STDMETHOD=UNCHANGED, the reconciled standard error $\tilde{\sigma}_{i,t}$ of $\tilde{x}_{i,t}$ is equal to the original standard error $\hat{\sigma}_{i,t}$ of $\hat{x}_{i,t}$.

When STDMETHOD=DISAGG, the reconciled standard error is proportional to the original disaggregated standard error and is computed as follows:

$$\tilde{\sigma}_{i,t} = w\hat{\sigma}_{i,t}$$

where $w = \frac{\tilde{x}_{i,t}}{\hat{x}_{i,t}}$.

When STDMETHOD=AGG, the reconciled standard error of $\tilde{x}_{i,t}$ is proportional to the aggregated standard error. When AGGREGATE=TOTAL, it is

$$\tilde{\sigma}_{i,t} = \hat{p}_{i,t}\hat{\sigma}_t$$

and when AGGREGATE=AVERAGE, it is

$$\tilde{\sigma}_{i,t} = \hat{p}_{i,t} m \hat{\sigma}_t$$

where $\hat{p}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^{m} \hat{x}_{j,t}}$, and $\hat{\sigma}_t$ is the standard deviation of $\hat{y}_t$.

If the selected method for the standard errors fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if STDMETHOD=DISAGG and the standard error is missing in the DISAGGDATA= data set, STDMETHOD=AGG is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the standard error was not reconciled according to your specification.

Confidence Limits

When CLMETHOD=SHIFT, the reconciled confidence limits are computed by recentering the original confidence limits around the reconciled predicted values.

When CLMETHOD=GAUSS, the reconciled confidence limits are computed assuming that the series is Gaussian with standard error equal to the reconciled standard error.

If the selected method for the confidence limits fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if CLMETHOD=SHIFT and the confidence limits are missing in the DISAGGDATA= data set, STDMETHOD=GAUSS is used instead. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the confidence limits were not reconciled according to your specification.

Bottom-Up Reconciliation

The goal of bottom-up (BU) reconciliation is to adjust $\hat{y}_t$ to obtain a new series $\{\tilde{y}_t\}$ of reconciled forecasts so that $\{\tilde{y}_t\}$ satisfies the aggregation constraint. When AGGREGATE=TOTAL, this is done by setting

$$\tilde{y}_t = \sum_{i=1}^{m} \hat{x}_{i,t} \qquad t = 1, 2, \ldots$$

When AGGREGATE=AVERAGE, this is done by setting

$$\tilde{y}_t = \frac{1}{m} \sum_{i=1}^{m} \hat{x}_{i,t} \qquad t = 1, 2, \ldots$$

Because the bottom-up problem is exactly identified and admits a unique solution, additional constraints on $\tilde{y}_t$ specified in the CONSTRAINT= data set are either already satisfied by the solution or result in an infeasible problem that will be flagged by the _RECONSTATUS_ variable in the OUTFOR= data set.

Missing Predicted Values

When some of the predicted values $\hat{x}_{i,t}$ are missing, the missing values are replaced by the actual values $x_{i,t}$, if these are present. This is done to prevent bias between the aggregated and reconciled forecasts, which results from models in which missing values in the predictions are generated because of the presence of lagged variables. However, if all predicted values $\hat{x}_{i,t}$ are missing, then the reconciled predicted value $\tilde{y}_t$ will also be missing, even though the actual values $x_{i,t}$ might not be missing.

Standard Errors

When STDMETHOD=UNCHANGED, the reconciled standard error $\tilde{\sigma}_t$ of $\tilde{y}_t$ is equal to the original standard error $\hat{\sigma}_t$ of $\hat{y}_t$.

When STDMETHOD=AGG, the reconciled standard error is proportional to the original aggregated standard error and is computed as follows:

$$\tilde{\sigma}_t = w\hat{\sigma}_t$$

where $w = \frac{\tilde{y}_t}{\hat{y}_t}$.

If STDMETHOD=DISAGG, the reconciled standard error $\tilde{\sigma}_t$ is equal to the square root of the sum of the squares of the disaggregated standard errors when AGGREGATE=TOTAL, and to the square root of the average of the squares of the disaggregated standard errors when AGGREGATE=AVERAGE.

If the selected method for the standard errors fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if STDMETHOD=AGG and the standard error is missing in the AGGDATA= data set, STDMETHOD=DISAGG is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the standard error was not reconciled according to your specification.

Confidence Limits

When CLMETHOD=SHIFT, the reconciled confidence limits are computed by recentering the original confidence limits around the reconciled predicted values.

When CLMETHOD=GAUSS, the reconciled confidence limits are computed assuming that the series is Gaussian with standard error equal to the reconciled standard error.

If the selected method for the confidence limits fails, PROC HPFRECONCILE tries to use a different method and displays a warning message in the log. For example, if CLMETHOD=SHIFT and the confidence limits are missing in the AGGDATA= data set, STDMETHOD=GAUSS is used instead, if possible. In such a case, the _RECONSTATUS_ variable identifies the observation that was not reconciled according to your preferences. You can also use the ERRORTRACE=ID option to display a message in the log that identifies the ID values for which the confidence limits were not reconciled according to your specification.

Data Set Input/Output

DISAGGDATA= Data Set

The DISAGGDATA= data set contains the variable(s) specified in the BY statement, the variable in the ID statement (when this statement is specified), and the following variables:

NAME	Variable name
ACTUAL	Actual values
PREDICT	Predicted values
LOWER	Lower confidence limits
UPPER	Upper confidence limits
ERROR	Prediction errors
STD	Prediction standard errors

Typically, the DISAGGDATA= data set is generated by the OUTFOR= option of the HPFENGINE procedure. See Chapter 10, "The HPFENGINE Procedure," for more details.

You can specify custom names for the variables in the DISAGGDATA= data set by using the DISAGGDATA statement.

AGGDATA= Data Set

The AGGDATA= data set contains a subset or none of the variables specified in the BY statement, the time ID variable in the ID statement (when this statement is specified), and the following variables:

NAME	Variable name
ACTUAL	Actual values
PREDICT	Predicted values
LOWER	Lower confidence limits
UPPER	Upper confidence limits
ERROR	Prediction errors
STD	Prediction standard errors

Typically, the AGGDATA= data set is generated by the OUTFOR= option of the HPFENGINE procedure. See Chapter 10, "The HPFENGINE Procedure," for more details.

You can specify custom names for the variables in the AGGDATA= data set by using the AGGDATA statement.

CONSTRAINT= Data Set

The CONSTRAINT= data set specifies the constraints to be applied to the reconciled forecasts. It contains the BY variables for the level at which reconciled forecasts are generated. That is, it contains the AGGBY variables when DIRECTION=BU, and the variables specified in the BY statement when DIRECTION=TD. If the _NAME_ variable is present in the AGGDATA= and DISAGGDATA= data set, it must also be present in the CONSTRAINT= data set. Additionally, the CONSTRAINT= data set contains the variable in the ID statement (when this statement is specified), and the following variables:

EQUALITY	specifies an equality constraint for the predicted reconciled values.
UNLOCK	A flag that specifies whether the equality constraint should be strictly enforced. Admissible values are as follows:

0	The equality constraint is locked.
1	The equality constraint is unlocked.

LOWERBD	Lower bounds for the reconciled forecasts
UPPERBD	Upper bounds for the reconciled forecasts

Locked equality constraints are treated as constraints in the top-down optimization process, and therefore their value is honored. Unlocked equalities are instead treated as regular forecasts and, in general, are changed by the reconciliation process.

If the NOTSORTED option is specified in the BY statement, then any BY group in the CONSTRAINT= data set that is out of order with respect to the BY groups in the AGGDATA= or DISAGGDATA= data set is ignored without any error or warning message. If the NOTSORTED option is not specified, then the BY groups in the CONSTRAINT= data set must be in the same sorted order as the AGGBY groups in the AGGDATA= data set when DIRECTION=BU, and in the same sorted order as the BY groups in the DISAGGDATA= data set when DIRECTION=TD; otherwise processing stops at the first such occurrence of a mismatch.

OUTFOR= Data Set

When DIRECTION=TD, the OUTFOR= data set contains the variables in the DISAGGDATA= data set and the _RECONSTATUS_ variable.

When DIRECTION=BU and the AGGDATA= data set has been specified, the OUTFOR= data set contains the variables in the AGGDATA= data set and the _RECONSTATUS_ variable. Otherwise, the AGGDATA= data set contains the BY variables specified in the AGGBY statement, the time ID variable in the ID statement (when this statement is specified), and the following variables:

NAME	Variable name
ACTUAL	Actual values
PREDICT	Predicted values

LOWER	Lower confidence limits
UPPER	Upper confidence limits
ERROR	Prediction errors
STD	Prediction standard errors
RECONSTATUS	Reconciliation status

If the RECDIFF option of the HPFRECONCILE statement has been specified, the OUTFOR= data sets will also contain the following variable:

RECDIFF	Difference between the reconciled predicted value and the original predicted value.

The _RECONSTATUS_ variable contains a code that specifies whether the reconciliation has been successful or not. A corresponding message is also displayed in the log. You can use the ERRORTRACE= option to define the resolution at which the error and warning messages are displayed in the log. The _RECONSTATUS_ variable can take the following values:

0	Success
500	A locked equality constraint has been imposed
1000	ID value out of the range with respect to the START= and END= interval
2000	Insufficient data to reconcile
3000	Reconciliation failed for the predicted value. This implies that it also failed for the confidence limits and standard error.
4000	Reconciliation failed for the standard error.
5000	Reconciliation failed for the confidence limits.
6000	The constrained optimization problem is infeasible.
7000	The option DISAGGREGATION=PROPORTION has been changed to DISAGGREGATION=DIFFERENCE for this observation because of a discordant sign in the input.
8000	The option STDMETHOD= provided by the user has been changed for this observation.
9000	The option CLMETHOD= provided by the user has been changed for this observation.
10000	The standard error hit the limits imposed by the STDDIFBD= option.
11000	Multiple warnings have been displayed in the log for this observation.

Examples

Example 15.1. Reconciling a Hierarchical Tree

The HPFRECONCILE procedure reconciles forecasts between two levels of a hierarchy. It can also be used recursively for reconciling the whole hierarchy.

Consider the hierarchy structure for the SASHELP.PRICEDATA data set outlined in Figure 15.1. You can reconcile the hierarchy top down, starting from the top level 0 down to the bottom level 2. At each new iteration, the OUTFOR= data set of the previous reconciliation step becomes the AGGDATA= data set of the current step.

First you need to compute the statistical forecasts for all levels. The statistical forecasts for level 1 and level 2 were already computed in the "Getting Started" section on page 471, so only the forecasts at the company levels are left to compute.

```
/*
/ Forecast series at company level
/-------------------------------------------------------------*/

*Step 1: model selection;
proc hpfdiagnose data=sashelp.pricedata
   outest=lvl0est
   modelrepository=work.mycat
   prefilter=both
   criterion=mape;
   id date interval=month notsorted;
   forecast sale / accumulate=total;
   input price / accumulate=average;
run;

*Step 2: estimation and forecasting;
proc hpfengine data=sashelp.pricedata inest=lvl0est
   out=_null_ outest=lvl0fest
   modelrepository=work.mycat outfor=lvl0for;
   id date interval=month notsorted;
   forecast sale / task=select accumulate=total;
   stochastic price /accumulate=average;
run;
```

First you reconcile the top and region levels. The output data set lvl1recfor contains the reconciled forecasts at level 1. This data set becomes the AGGDATA= data set for the next step of TD reconciliation that involves level 1 and level 2. You can check that the reconciled forecasts at level 2 add up to the forecasts at level 0.

```
/*
/ Reconcile forecasts top down from company to region
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl1for aggdata=lvl0for
     direction=TD
     outfor=lvl1recfor;
   id date interval=month;
   by region;
run;

/*
/ Reconcile forecasts top down from region to region/product
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2for aggdata=lvl1recfor
     direction=TD
     outfor=lvl2recfor;
   id date interval=month;
   by region product;
run;

/*
/ Verify that level 2 forecasts add up to level 0 forecasts
/-------------------------------------------------------------*/
proc timeseries data=lvl2recfor out=toprec ;
   id date interval=month notsorted  accumulate=total;
   var predict;
run;

proc compare base=lvl0for compare=toprec criterion=0.00001;
   var predict;
run;
```

You can also reconcile the hierarchy from the bottom up. In such a case, the OUTFOR= data set of the previous step becomes the DISAGGDATA= data set of the current step.

Alternatively, you could choose to reconcile the hierarchy from the *middle out* from an intermediate level. In this case, you choose an intermediate level as a starting point, and reconcile all levels above from the bottom up, while reconciling all levels below from the top down. In the following SAS code, the hierarchy of SASHELP.PRICEDATA is reconciled from the middle out, starting from level 1.

```
/*
/ Reconcile forecasts bottom up from region to company
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl1for aggdata=lvl0for
     direction=BU
     outfor=lvl0recfor;
   id date interval=month;
   by region;
run;

/*
/ Reconcile forecasts top down from region to region/product
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2for aggdata=lvl1for
     direction=TD
     outfor=lvl2recfor;
   id date interval=month;
   by region product;
run;
```

You can use the external forecasts feature of the HPFENGINE procedure to generate summary statistics and statistics of fit for the reconciled forecasts, as shown in the following SAS statements for the company level.

First, an external model spec is generated using PROC HPFEXMSPEC. The characteristics of estimated models that determine the options for PROC HPFEXMSPEC can be found in the OUTEST= data set of the HPFENGINE call for the corresponding level. In this case, the lvl0fest data set shows that the estimated model has three parameters and that the dependent variable sales has not undergone any transformation.

```
/*
/ Generate external model spec
/-------------------------------------------------------------*/
proc hpfexmspec modelrepository=work.mycat
   specname=lvl0exm;
   exm transform=none nparms=3;
run;
```

Subsequently, a selection list containing the external model is defined with PROC HPFSELECT.

```
/*
/ Generate select list
/-------------------------------------------------------------*/
proc hpfselect modelrepository=work.mycat
   selectname=lvl0selexm;
   spec lvl0exm;
run;
```

Finally, the EXTERNAL statement of the HPFENGINE procedure is used in conjunction with the FORECAST statement to generate the OUTSTAT= and OUTSUM= data sets that correspond to the reconciled forecasts input data set lvl0recfor and the model specifications contained in the external model lvl0exm.

```
/*
/ Create OUTSTAT= and OUTSUM= data sets
/-------------------------------------------------------------*/
proc hpfengine data=lvl0recfor(rename=(actual=sales))
      out=_NULL_
      outstat=lvl0outstat
      outsum=lvl0outsum
      modelrepository=work.mycat
      globalselection=lvl0selexm;
   id date interval=month notsorted;
   forecast sales;
   external sales=(predict=predict lower=lower
      upper=upper stderr=std);
run;
```

Example 15.2. Aggregating Forecasts

If you do not provide the AGGDATA= input data set, but provide only the DISAGGDATA= data set, PROC HPFRECONCILE aggregates the forecasts according to the BY variable that you specify in the AGGBY option. If you use the options STDMETHOD=DISAGG and CLMETHOD=GAUSS, you can obtain standard errors and confidence interval as well.

In this example, the forecasts at level 2 of Figure 15.1 are aggregated to find forecasts at level 1 for the SASHELP.PRICEDATA data set.

```
/*
/ Aggregate region/product forecasts to region level
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2for
     direction=BU
     outfor=lvl1aggfor
     stdmethod=disagg
     clmethod=gauss;
   id date interval=month;
   by region product;
   aggby region;
run;
```

Example 15.3. Disaggregating Forecasts

You can use the HPFRECONCILE procedure to disaggregate top-level forecasts according to proportions that you supply. This can be accomplished by creating a DISAGGDATA= data set that contains the proportions that you want to use in place of the PREDICT variable.

In this example, the level 1 forecasts of the variable sale in the SASHELP.PRICEDATA data set are disaggregated to level 2 according to the historical median proportions.

First, a combination of DATA steps and PROC UNIVARIATE is used to compute the median proportions and merge them with the level 2 OUTFOR= data set from PROC HPFENGINE.

```
/*
/ Compute total sales per region
/-------------------------------------------------------------*/

proc timeseries data=sashelp.pricedata out=lvl1sales ;
   id date interval=month notsorted  accumulate=total;
   by region;
   var sale;
run;
```

```
/*
/ Compute sale proportions
/-------------------------------------------------------------*/

proc sort data=sashelp.pricedata out=tmp;
   by region date;
run;

data lvl2prop;
   merge tmp lvl1sales(rename=(sale=totsale));
   by region date;
   prop = sale / totsale;
run;



/*
/ Compute median sale proportions
/-------------------------------------------------------------*/

proc sort data=lvl2prop;
   by region product;
run;

proc univariate data=lvl2prop noprint;
   var prop;
   by region product;
   output out=lvl2medprop median=medprop;
run;



/*
/ Merge median proportions with level2 OUTFOR
/-------------------------------------------------------------*/
data lvl2medfor;
   merge lvl2for lvl2medprop;
   by region product;
run;
```

Then PROC HPFRECONCILE is invoked, using the DISAGGDATA statement to specify that the variable medprop is to be used instead of the default PREDICT.

Note that the proportions need not sum to one. PROC HPFRECONCILE automatically rescales them to sum to one.

```
/*
/ Disaggregate level1 forecasts according to median sale
/-------------------------------------------------------------*/

proc hpfreconcile disaggdata=lvl2medfor aggdata=lvl1for
      direction=TD
      stdmethod=unchanged
      clmethod=gauss
      outfor=lvl2recmedfor;
   disaggdata predict=medprop;
   by region product;
run;
```

The variable medprop in the OUTFOR=lvl2recmedfor data set contains the disaggregated forecasts according to the proportions that you supplied.

In this case the options STDMETHOD=UNCHANGED and CLMETHOD=GAUSS have been used to obtain standard errors and confidence intervals. However, you need to be aware that they might not be reliable.

Alternatively, if you are interested in disaggregating the predicted values only, you can use the PREDICTONLY option as in the following code.

```
/*
/ Disaggregate level1 predict only
/---------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2medfor aggdata=lvl1for
      direction=TD
      predictonly
      outfor=lvl2recmedfor;
   disaggdata predict=medprop;
   by region product;
run;
```

Example 15.4. Imposing Constraints

You can impose constraints on the reconciled forecasts by using the CONSTRAINT= option or the SIGN= option.

In this example, you revisit Example 15.1 and impose different types of constraints on the reconciled forecasts. Suppose you want all reconciled forecasts to be nonnegative, and for the month of April 2003 you want the following:

1. Product 1 at Region 1 to have a locked equality of 400
2. Product 2 at Region 1 to have an unlocked equality of 400
3. Product 4 at Region 2 to be less or equal to 300

First you need to create a CONSTRAINT= data set that contains the constraints you want for the date of April 2003.

```
/*
/ Create constraint data set
/-------------------------------------------------------------*/
data constraint;
   length _name_ $32;
   input region product _name_ $ date MONYY7. equality
         unlock lowerbd upperbd;
   datalines;
   1 1 sale Apr2003 400 0 . .
   1 2 sale Apr2003 400 1 . .
   2 4 sale Apr2003 . .   . 300
   ;
run;
```

Then you reconcile the two levels by using the SIGN=NONNEGATIVE option to impose the nonnegativity constraint, and by using the CONSTRAINT= option to impose your constraints on the reconciled forecasts in April 2003. The PREDICTONLY option of the HPFRECONCILE statement restricts the reconciliation to the PREDICT variable.

```
/*
/ Reconcile forecasts with constraints
/-------------------------------------------------------------*/
proc hpfreconcile disaggdata=lvl2for aggdata=lvl1for
     direction=TD
     sign=nonnegative
     constraint=constraint
     outfor=lvl2recfor
     predictonly;
   id date interval=month;
   by region product;
run;
```

Chapter 16
The HPFSELECT Procedure

Chapter Contents

Chapter 16
The HPFSELECT Procedure

Overview

The HPFSELECT procedure enables you to control the forecasting model selection process by defining lists of candidate forecasting models. Using model selection lists created by PROC HPFSELECT, you can control which forecasting model or models SAS High-Performance Forecasting uses to forecast particular time series.

The HPFSELECT procedure creates model selection files and stores them in a repository for later use by the HPFENGINE procedure. These model selection files list model specifications previously created and stored in a model repository by the HPFARIMASPEC, HPFESMSPEC, HPFEXMSPEC, HPFIDMSPEC, or HPFUCMSPEC procedure.

Using PROC HPFSELECT, you can also specify other options that control the forecasting model selection process.

Getting Started

The following example shows how to create a model selection list file. Suppose the model repository MYLIB.MYMODELS contains four model specification files (A.XML, B.XML, C.XML, and D.XML), and you want to create a model selection list that will tell PROC HPFENGINE to automatically select between these models based on the mean absolute percentage error (MAPE). The following SAS statements accomplish this.

```
proc hpfselect  repository=mylib.mymodels
                name=myselect
                label="My model selection list";
  spec a b c d;
  select criterion=mape;
run;
```

The options in the PROC HPFSELECT statement specify the name and location of the model selection file that is created. The REPOSITORY= option specifies that the output file be placed in a catalog MYLIB.MYMODELS, the NAME= option specifies that the name of the file be “myselect.xml”, and the LABEL= option specifies a descriptive label for the selection list MYSELECT. The SPEC statement specifies the list of candidate models. These model specifications must also be stored in MYLIB.MYMODELS. The SELECT statement specifies options that control how PROC HPFENGINE selects among the candidate models when applying the selection list MYSELECT to actual time series data.

Syntax

The following statements are used with the HPFSELECT procedure.

PROC HPFSELECT *options*;
  DIAGNOSE *options*;
  FORECASTOPTIONS *options*;
  SELECT *options*;
  SPECIFICATION *specification-list / options*;

Functional Summary

The statements and options controlling the HPFSELECT procedure are summarized in the following table.

Description	Statement	Option
Statements		
specifies the forecasting options	FORECASTOPTIONS	
Model Repository Options		
specifies the model repository	PROC HPFSELECT	REPOSITORY=
specifies the model specification name	PROC HPFSELECT	NAME=
specifies the model specification label	PROC HPFSELECT	LABEL=
Forecasting Options		
specifies the confidence limit width	FORECASTOPTIONS	ALPHA=
Model Selection Options		
specifies the forecast holdout sample size	SELECT	HOLDOUT=
specifies the forecast holdout sample percent	SELECT	HOLDOUTPCT=
specifies the model selection criterion	SELECT	CRITERION=
Model Specification Options		
maps specification symbol to data set variable	SPECIFICATIONS	INPUTMAP
adds event to specification	SPECIFICATIONS	EVENTAP
associate external data with specification	SPECIFICATIONS	EXMMAP
associate external subroutine with specification	SPECIFICATIONS	EXMFUNC
override specification labels	SPECIFICATIONS	LABEL

Description	Statement	Option
Diagnostic Options		
specifies the intermittency test threshold	DIAGNOSE	INTERMITTENT=
specifies the seasonality test	DIAGNOSE	SEASONTEST=

PROC HPFSELECT Statement

PROC HPFSELECT *options ;*

The following options can be used in the PROC HPFSELECT statement.

LABEL= *SAS-label*
specifies a descriptive label for the model selection to be stored in the SAS catalog or directory. The LABEL= option can also be specified as SELECTLABEL=.

NAME= *SAS-name*
names the model selection file to be stored in the SAS catalog or directory. The NAME= option can also be specified as SELECTNAME=.

REPOSITORY= *SAS-catalog-name* | *SAS-file-reference*
names the SAS catalog or directory to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

DIAGNOSE Statement

DIAGNOSE *options ;*

The DIAGNOSE statement is used to specify diagnostic options.

The following examples illustrate typical uses of the DIAGNOSE statement:

```
/* same as default options */
diagnose intermittent=2.0 seasontest=(siglevel=0.01);

/* no seasonality */
diagnose seasontest=(siglevel=0);
```

INTERMITTENT= *number*
specifies a number greater than one that is used to determine whether or not a time series is intermittent. If the average demand interval is greater than this number, then the series is assumed to be intermittent. The default is INTERMITTENT=2.0.

SEASONTEST= *option*
specifies the options related to the seasonality test.

The following values for the SEASONTEST= options are allowed:

NONE	No test.
(SIGLEVEL=number)	Significance probability value to use in testing whether seasonality is present in the time series. The value must be between 0 and 1.

A smaller value of the SIGLEVEL= option means that stronger evidence of a seasonal pattern in the data is required before PROC HPFENGINE will use seasonal models to forecast the time series. The default is SEASONTEST=(SIGLEVEL=0.01).

FORECASTOPTIONS Statement

FORECASTOPTIONS *options;*

The FORECASTOPTIONS statement is used to specify forecasting options.

ALPHA= *number*

specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA=value must be between 0 and 1. The default is ALPHA=0.05, which produces 95% confidence intervals.

SELECT Statement

SELECT *options;*

The SELECT statement is used to specify model selection options.

The following examples illustrate typical uses of the SELECT statement:

```
/* same as default options */
select criterion=rmse holdout=0 holdoutpct=0;

/* selection criterion mape with absolute holdout size 6 */
select criterion=mape holdout=6;
```

HOLDOUT= *n*

specifies the size of the holdout sample to be used for model selection. The holdout sample is a subset of actual time series ending at the last nonmissing observation. The default is zero (no holdout sample).

HOLDOUTPCT= *number*

specifies the size of the holdout sample as a percentage of the length of the time series. If HOLDOUT=5 and HOLDOUTPCT=10, the size of the holdout sample is $min(5, 0.1T)$ where T is the length of the time series with beginning and ending missing values removed. The default is 100 (100%), which means no restriction on the holdout sample size based on the series length.

CRITERION= *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option would often be used in conjunction with the

HOLDOUT= option. The default is CRITERION=RMSE. The following is the list of valid values for the CRITERION= option and the statistics of fit these option values specify:

SSE	Sum of Square Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
UMSE	Unbiased Mean Square Error
URMSE	Unbiased Root Mean Square Error
MAXPE	Maximum Percent Error
MINPE	Minimum Percent Error
MPE	Mean Percent Error
MAPE	Mean Absolute Percent Error
MDAPE	Median Absolute Percent Error
GMAPE	Geometric Mean Absolute Percent Error
MINPPE	Minimum Predictive Percent Error
MAXPPE	Maximum Predictive Percent Error
MPPE	Mean Predictive Percent Error
MAPPE	Symmetric Mean Absolute Predictive Percent Error
MDAPPE	Median Absolute Predictive Percent Error
GMAPPE	Geometric Mean Absolute Predictive Percent Error
MINSPE	Minimum Symmetric Percent Error
MAXSPE	Maximum Symmetric Percent Error
MSPE	Mean Symmetric Percent Error
SMAPE	Symmetric Mean Absolute Percent Error
MDASPE	Median Absolute Symmetric Percent Error
GMASPE	Geometric Mean Absolute Symmetric Percent Error
MINRE	Minimum Relative Error
MAXRE	Maximum Relative Error
MRE	Mean Relative Error
MRAE	Mean Relative Absolute Error
MDRAE	Median Relative Absolute Error
GMRAE	Geometric Mean Relative Absolute Error
MAXERR	Maximum Error
MINERR	Minimum Error
ME	Mean Error
MAE	Mean Absolute Error

RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion

SPECIFICATIONS Statement

SPECIFICATIONS *specification-list / options;*

The SPECIFICATIONS statement is used to list model specifications. There can be any number of models specifications in the list and any number of SPECIFICATIONS statements.

The model specifications must be contained in the model repository specified by the REPOSITORY= option of the HPFSELECT procedure statement.

The following options can be used with the SPECIFICATIONS statement.

INPUTMAP (*SYMBOL=***string** *VAR=***variable)***;*

associates the symbols in a model specification with variable names in a DATA= data set.

The SYMBOL= option should match a symbol defined in a model specification. The VAR= option should match a variable name in a data set. When the model selection list is used in conjunction with the HPFENGINE procedure, the PROC HPFENGINE statement DATA= option specifies the input data set that contains the variable.

Mappings are needed because model specifications are generic. For example, suppose a model specification associates the symbol Y with the dependent variable. If you want to use this model to forecast the variable OZONE, you map Y to OZONE with INPUTMAP(SYMBOL=Y VAR=OZONE). If you later want to use the same specification to forecast SALES, you map Y to SALES with INPUTMAP(SYMBOL=Y VAR=SALES).

The INPUTMAP option is not required. By default, PROC HPFENGINE attempts to locate variables in its DATA= data set that match the symbols in each specification listed in the selection list.

EVENTMAP (*SYMBOL=***_NONE_** *EVENT=***eventDef <NODIFF>)**
EVENTMAP (*SYMBOL=***string** *EVENT=***eventDef)**

associates events with a model specification.

If SYMBOL=_NONE_ is used, the event specified in eventDef is added to the model as a simple regressor. By default, for an ARIMA model, any differencing applied to the dependent variable is applied in the same manner to the new event input. Specifying NODIFF indicates no differencing should be performed on the event.

If the SYMBOL string matches one of the symbols specified as input in either a UCM or ARIMA model, the event data will be used for the matching input. In this manner, events can enter models through complex transfer functions. The NODIFF option does not apply in this case, since differencing will be explicitly described in the input statement of the model.

If the event referenced by eventDef is a "predefined event" (see the HPFEVENT procedure) documentation, no INEVENT= option is required for the HPFENGINE procedure. Otherwise, the event named must be defined in an event data set by using the HPFEVENT procedure, and that data set must be given to the HPFENGINE procedure with the INEVENT= option. An error will occur during PROC HPFENGINE model selection if eventDef is not found in an event data set.

Only UCM and ARIMA models are valid when EVENTMAP is used. If another model type, such as exponential smoothing, has an EVENTMAP option, the option is simply ignored.

EXMMAP*(options)***;**

associates an external model specification with variable names in a DATA= data set, thus identifying the source of forecasts and optionally prediction standard errors, and the lower and upper confidence limits.

Available options are as follows:

PREDICT= var

identifies the variable to supply forecasts and is required.

STDERR= var

identifies the variable to supply the prediction standard error.

LOWER= var

identifies the variable to supply the lower confidence limit.

UPPER= var

identifies the variable to supply the upper confidence limit.

For example, if you want an external model "myexm" to use forecasts from the variable "yhat" in the DATA= data set passed to the HPFENGINE procedure, the appropriate statement would be as follows:

```
spec myexm / exmmap(predict=yhat);
```

If you also want to use prediction standard errors from the "std" variable in the same data set, use the following statement:

```
spec myexm / exmmap(predict=yhat stderr=std);
```

EXMFUNC*(string)***;**

associates an external model specification with a user-defined subroutine. The string parameter is the signature of the subroutine and has the following form:

subroutineName(parameters)

where parameters indicate the order, number, and type of arguments in the user-defined subroutine. The parameter _PREDICT_ is required, indicating the return of forecast values. Optional parameters for return of other data from the user-defined subroutine include the following:

STDERR	Prediction standard error
LOWER	Lower confidence limit
UPPER	Upper confidence limit

The HPFENGINE procedure can also pass data into the user-defined subroutine by using the following options:

TIMEID	Time ID
SEASON	Seasonal index
ACTUAL	Actual values

As an example, suppose the signature of a user-defined subroutine is defined in the FCMP procedure as follows:

```
subroutine userdef1(act[*], pred[*]);
```

Also suppose this subroutine is mapped to the external model “myexm” by using the following statement:

```
spec myexm / exmfunc('userdef1(_actual_ _predict_ )');
```

Then the HPFENGINE procedure will pass the array of actuals to the subroutine userdef1, and the function will compute the forecasts and return them to the HPFENGINE procedure.

Next, consider the case where a user-defined subroutine requires actuals, time ID values, and seasonal indices in order to compute and return forecasts and prediction standard errors. The subroutine might be defined as follows:

```
subroutine complexsub(act[*], timeid[*],
                      seasons[*], pred[*], stderr[*]);
```

It would be mapped to an external model named “myexm” by using the following statement:

```
spec myexm / exmfunc(
     'complexsub(_actual_ _timeid_ _season_ _predict_ _stderr_)'
                    );
```

This syntax is demonstrated further in Example 4.

LABEL= *SAS-label*

overrides the label in the model specification and causes the new label to print in the model selection list in the HPFENGINE procedure. This option is useful if you have the same model specification listed more than once and want to distinguish between them in the HPFENGINE procedure output.

As an example, consider the following case of a single external model used twice in the selection list, with each occurrence mapped to a different external forecast:

```
spec myexm / exmmap(predict=yhatRALEIGH)
             label="External Model: Raleigh Forecasts";
spec myexm / exmmap(predict=yhatATLANTA)
             label="External Model: Atlanta Forecasts"
```

In the model selection list output from the HPFENGINE procedure, the new labels appear, rather than the label in “myexm” repeated twice.

Examples

Example 16.1. The INPUTMAP Option

Consider the following example. The HPFUCMSPEC procedure is used to define a UCM spec. The dependent variable is assigned the symbol Y, and the three inputs are assigned the symbols X1, X2, and X3, respectively. You ultimately want to forecast the series contained in the variable OZONE with inputs X1, SUMMER, and WINTER. The INPUTMAP option in PROC HPFSELECT is used to tell PROC HPFENGINE that OZONE should replace Y, SUMMER should replace X2, and WINTER should replace X3. The default behavior occurs for X1.

```
proc hpfucmspec repository=sasuser.mycat
                name=myucm
                label="My UCM spec";
   dependent symbol=Y;
   input symbol=X1;
   input symbol=X2;
   input symbol=X3;
   level;
   season length=12 type=trig;
run;

proc hpfselect repository=sasuser.mycat
                name=myselect
                label="My Selection List";
   select criterion=rmse holdout=12;
   spec myucm /
   inputmap(symbol=Y var=OZONE)
   inputmap(symbol=X2 var=SUMMER)
   inputmap(symbol=X3 var=WINTER);
run;
```

The same result could be achieved by making the symbol in the model specification match the variables in the HPFENGINE procedure's DATA= data set.

```
proc hpfucmspec repository=sasuser.mycat
                name=myucm
                label="My UCM spec";
     dependent symbol=OZONE;
     input symbol=X1;
     input symbol=SUMMER;
     input symbol=WINTER;
     level;
     season length=12 type=trig;
run;

proc hpfselect repository=sasuser.mycat
                name=myselect
                label="My Selection List";
     select criterion=rmse holdout=12;
     spec myucm;
run;
```

The obvious disadvantage here is that the model specification and data set are tightly linked.

Example 16.2. The EVENTMAP Option

Events are dynamically added as simple regressors to UCM and ARIMA models by using the EVENTMAP option in the SPECIFICATIONS statement.

You first create an ARIMA model and create a selection list that directs the HPFENGINE procedure to choose between this model without an event and this model with the event. The output is shown in Output 16.2.1.

```
    proc hpfevents data=sashelp.air;;
       eventdef summer = (june july august);
       eventdata out=eventDB;
    run;

    proc hpfarimaspec modelrepository=work.repo name=arima;
       forecast symbol=air q=(1 12) transform=log;
    run;

    proc hpfselect modelrepository=work.repo name=select;
       spec arima;
       spec arima / eventmap(symbol=_none_ event=summer);
    run;
```

```
title1 "HPFENGINE dynamically adds event from event database";
proc hpfengine data=sashelp.air modelrepository=work.repo outest=outest1
      globalselection=select print=(select estimates) inevent=eventDB;
   id date interval=month;
   forecast air;
run;
```

Output 16.2.1. Selection and Estimation Results

```
              HPFENGINE dynamically adds event from event database

                            The HPFENGINE Procedure

                        Model Selection Criterion = MAPE

     Model      Statistic     Selected     Label

     ARIMA      20.048903     No           ARIMA: Log( AIR ) ~  Q = (1,12)
     ARIMA      19.654381     Yes          ARIMA: Log( AIR ) ~  Q = (1,12)


                              Parameter Estimates

                                                Standard                Approx
 Component     Parameter       Estimate            Error     t Value    Pr > |t|

 AIR           CONSTANT         5.47071          0.04202      130.21      <.0001
 AIR           MA1_1           -0.80089          0.02337      -34.27      <.0001
 AIR           MA1_12          -0.27008          0.02374      -11.37      <.0001
 SUMMER        SCALE            0.15504          0.05859        2.65      0.0091
```

You calculate the same results but this time explicitly create a model that includes the input in its specification. Then the event is added to the data set. Note that the results, shown in Output 16.2.2, are the same as the results of the simpler usage with EVENTMAP.

```
data air(keep=date air summer);
   set sashelp.air;
   summer = 0;
   if month(date) eq 6 or
      month(date) eq 7 or
      month(date) eq 8 then summer = 1;
run;

proc hpfarimaspec modelrepository=work.repo name=arimasummer;
   forecast symbol=air q=(1 12) transform=log;
   input symbol=summer;
run;

proc hpfselect modelrepository=work.repo name=select;
   spec arima arimasummer;
run;


title1 "HPFENGINE uses input present in data set and specified in model";
proc hpfengine data=air modelrepository=work.repo outest=outest2
```

```
      globalselection=select print=(select estimates);
   id date interval=month;
   forecast air;
   input summer;
run;
```

Output 16.2.2. Selection and Estimation Results

```
        HPFENGINE uses input present in data set and specified in model

                           The HPFENGINE Procedure

                      Model Selection Criterion = MAPE

                     Model            Statistic    Selected

                     ARIMA            20.048903    No
                     ARIMASUMMER      19.654381    Yes

                       Model Selection Criterion = MAPE

       Model          Label

       ARIMA          ARIMA: Log( AIR ) ~  Q = (1,12)
       ARIMASUMMER    ARIMA: Log( AIR ) ~  Q = (1,12)   +  INPUT:  SUMMER


                             Parameter Estimates

                                             Standard                Approx
   Component    Parameter      Estimate         Error     t Value    Pr > |t|

   AIR          CONSTANT        5.47071       0.04202      130.21      <.0001
   AIR          MA1_1          -0.80089       0.02337      -34.27      <.0001
   AIR          MA1_12         -0.27008       0.02374      -11.37      <.0001
   summer       SCALE           0.15504       0.05859        2.65      0.0091
```

Example 16.3. The DIAGNOSE Statement

The DIAGNOSE statement allows control of the diagnostics used by PROC HPFENGINE to subset the model selection list. Two ESM model specifications are created in this example, one seasonal and one nonseasonal. They are placed together in a selection list with the seasonality test value allowed to remain at its default in this first case. The diagnostics in PROC HPFENGINE judge the dependent series as seasonal and therefore exclude the nonseasonal model from consideration. The output is shown in Output 16.3.1.

```
proc hpfesmspec modelrepository=work.repo name=logdouble;
   esm method=double transform=log;
run;

proc hpfesmspec modelrepository=work.repo name=logwinters;
   esm method=winters transform=log;
run;

proc hpfselect modelrepository=work.repo name=select;
```

```
   spec logdouble logwinters;
run;


title1 "Use default seasonality test option";
proc hpfengine data=sashelp.air modelrepository=work.repo outest=outest1
      globalselection=select print=all;
   id date interval=month;
   forecast air;
run;
```

Output 16.3.1. Seasonality Test Significance Level of 0.01

```
                      Use default seasonality test option

                            The HPFENGINE Procedure

                       Model Selection Criterion = MAPE

Model          Statistic    Selected    Label

LOGDOUBLE       .           Removed     Log Double Exponential Smoothing
LOGWINTERS     2.7138783    Yes         Log Winters Method (Multiplicative)
```

The same two exponential smoothing models are used in a selection list again, but this time the seasonality test is disabled. Both models are fit to the series as a result. The output is shown in Output 16.3.2.

```
proc hpfselect modelrepository=work.repo name=select;
   spec logdouble logwinters;
   diagnose seasontest=none;
run;


title1 "Turn off the seasonality test";
proc hpfengine data=sashelp.air modelrepository=work.repo outest=outest1
      globalselection=select print=select;
   id date interval=month;
   forecast air;
run;
```

Output 16.3.2. No Seasonality Test Performed

```
                         Turn off the seasonality test

                            The HPFENGINE Procedure

                       Model Selection Criterion = MAPE

Model          Statistic    Selected    Label

LOGDOUBLE      10.167638    No          Log Double Exponential Smoothing
LOGWINTERS      2.713878    Yes         Log Winters Method (Multiplicative)
```

Finally, the seasonality test significance level is set to zero so that the series will not be judged as seasonal. In Output 16.3.3, note that the nonseasonal model is fit to the series, but the seasonal model is removed.

```
proc hpfselect modelrepository=work.repo name=select;
   spec logdouble logwinters;
   diagnose seasontest=(siglevel=0);
run;
```

```
title1 "Set significance level so that series is not judged as seasonal";
proc hpfengine data=sashelp.air modelrepository=work.repo outest=outest1
      globalselection=select print=select;
   id date interval=month;
   forecast air;
run;
```

Output 16.3.3. All Series Treated as Nonseasonal

```
      Set significance level so that series is not judged as seasonal

                             The HPFENGINE Procedure

                        Model Selection Criterion = MAPE

Model          Statistic    Selected    Label

LOGDOUBLE      10.167638    Yes         Log Double Exponential Smoothing
LOGWINTERS       .          Removed     Log Winters Method (Multiplicative)
```

Example 16.4. External Models and User-Defined Subroutines

The EXMFUNC options enables you to associate an external model specification with a user-defined subroutine. First, define a user subroutine and store it in a catalog. In this case, you create a simple three-point moving average.

```
proc fcmp outlib=sasuser.hpfengine.funcs;
   subroutine move_avg3(act[*], pred[*]);
      actlen = DIM(act);
   predlen = DIM(pred);
   pred[1] = 0;
   pred[2] = act[1]/3.0;
   pred[3] = (act[1] + act[2])/3.0;
   do i=4 to actlen+1;
      pred[i] = (act[i-1] + act[i-2] + act[i-3])/3.0;
   end;
   do i=actlen+2 to predlen;
      pred[i] = (pred[i-1] + pred[i-2] + pred[i-3])/3.0;
   end;
   endsub;
run;
options cmplib = sasuser.hpfengine;
```

Now, just for comparison, use the HPFARIMASPEC procedure to make a model specification that will produce the same three-point moving average.

```
proc hpfarimaspec modelrepository=work.repo name=arima;
   forecast symbol=y noint p=3
      ar=(0.333333333 0.333333333 0.333333333);
   estimate noest;
run;
```

Next, use the HPFEXMSPEC procedure to create an external model specification and the HPFSELECT procedure to make a selection list with the external model, referencing the user-defined subroutine, and the ARIMA model.

```
proc hpfexmspec modelrepository=work.repo name=myexm;
   exm;
run;

proc hpfselect modelrepository=work.repo name=select;
   diagnose seasontest=none;
   spec arima;
   spec myexm / exmfunc('move_avg3(_actual_ _predict_ )')
                label="External Model from move_avg3";
run;
```

Finally, use the HPFENGINE procedure to forecast a series by using the selection list just created. As expected, both models produce the same forecast, as indicated by the same selection fit statistic.

```
proc hpfengine data=sashelp.air out=_null_ modelrepository=work.repo
      globalselection=select print=select;
   id date interval=month;
   forecast air;
run;
```

The output is shown in Output 16.4.1.

Output 16.4.1. External Model with User-Defined Subroutine vs. ARIMA Model

```
                         The HPFENGINE Procedure

                    Model Selection Criterion = MAPE

   Model     Statistic     Selected     Label

   ARIMA     13.455362     No           ARIMA:  Y  ~ P = 3    NOINT
   MYEXM     13.455362     Yes          External Model from move_avg3
```

Example 16.5. Comparing Forecasts from Multiple External Sources

The EXMMAP option enables you to associate an external model specification with variables in a data set. This example uses the EXPAND procedure and the ARIMA procedure to generate data for the external forecasts. In practice this external data might come from judgmental forecasts or other systems. The external forecasts must be in the same data set as the series you are modeling.

```
data temp;
   set sashelp.air;
   drop i;
* introduce some missing for EXPAND to fill in;
   if mod(_n_, 5) eq 0 them air = .;
   if mod(_n_+1, 5) eq 0 them air = .;
   if mod(_n_+2, 5) eq 0 them air = .;
   output;

   if date eq '01dec1960'd then do;
      do i=1 to 4;
         date = intnx('month', '01dec1960'd, i);
         air = .;
         output;
      end;
   end;
run;

proc expand data=temp extrapolate out=expandout(rename=(air=interp));
   id date;
   convert air;
run;

data temp;
   set sashelp.air;
   logair = log(air);
run;

proc arima data=temp;
   identify var=logair(1,12) noprint;
   estimate q=(1)(12) noconstant method=ml noprint;
   forecast out=arimaout(rename=(forecast=airline)) lead=4 id=date
      interval=month noprint;
quit;

data arimaout;
   set arimaout;
   airline = exp(airline);
run;

data temp;
   keep date interp airline air;
   merge expandout arimaout sashelp.air;
   by date;
run;
```

Next, create a smoothing model to add to the selection, demonstrating that you can compare multiple external forecasts not only with one another but also with other statistical models. After the models are defined, they are added to a selection list. Notice that the same external model can be associated with different external forecasts and that the LABEL= option can be used to help differentiate between the two. Finally, the HPFENGINE procedure is called, and you see that the forecast originally generated by the ARIMA procedure best fits the historical data.

```
proc hpfesmspec modelrepository=work.repo name=esm;
   esm method=seasonal transform=auto;
run;

proc hpfexmspec modelrepository=work.repo name=exm;
   exm;
run;

proc hpfselect modelrepository=work.repo name=select;
   spec esm;
   spec exm / exmmap(predict=interp) label="Interpolation from PROC EXPAND";
   spec exm / exmmap(predict=airline) label="Forecasts from PROC ARIMA";
run;

proc hpfengine data=temp out=_null_ modelrepository=work.repo
      globalselection=select print=select lead=4;
   id date interval=month;
   forecast air;
   external interp airline;
run;
```

The output is shown in Output 16.5.1.

Output 16.5.1. Two External Forecasts and a Smoothing Model

```
                          The HPFENGINE Procedure

                           Variable Information

    Name                                                                AIR
    Label                               international airline travel (thousands)
    First                                                           JAN1949
    Last                                                            DEC1960
    Number of Observations Read                                         148


                     Model Selection Criterion = MAPE

     Model     Statistic     Selected     Label

     ESM       3.1909590     No           Log Seasonal Exponential Smoothing
     EXM       5.5023045     No           Interpolation from PROC EXPAND
     EXM       2.9239173     Yes          Forecasts from PROC ARIMA
```

Example 16.6. Input to User-Defined Subroutines

To illustrate the different type of data that the HPFENGINE procedure can pass to a user-defined subroutine, consider the following subroutine definition:

```
proc fcmp outlib=sasuser.hpfengine.funcs;
   subroutine testsub(timeid[*], act[*], seasons[*],  pred[*]);
   actlen = DIM(act);
   predlen = DIM(pred);
   format date monyy.;

   * print the input;
   do i=1 to 24;
      date = timeid[i]; actual=act[i]; season=seasons[i];
      put i= date= actual= season=;
   end;

   * just return mean;
   mean = 0.0;
   do i=1 to actlen;
      mean = mean + act[i];
   end;
   mean = mean / actlen;

   do i=1 to predlen;
      pred[i] = mean;
   end;

   endsub;
run;
```

Suppose you have an external model specification and invocation of the HPFSELECT procedure such as the following:

```
proc hpfselect modelrepository=work.repo name=select;
   diagnose seasontest=none;
   spec myexml / exmfunc('testsub(_timeid_ _actual_ _season_ _predict_)');
run;
```

Then suppose you have the following call to the HPFENGINE procedure to forecast the series AIR in SASHELP.AIR:

```
proc hpfengine data=sashelp.air out=_null_ outfor=outfor
      modelrepository=work.repo globalselection=select;
   id date interval=month;
   forecast air;
run;
```

The output would look like the following:

```
i=1  date=JAN49 actual=112 season=0
i=2  date=FEB49 actual=118 season=1
i=3  date=MAR49 actual=132 season=2
i=4  date=APR49 actual=129 season=3
i=5  date=MAY49 actual=121 season=4
```

```
i=6  date=JUN49 actual=135 season=5
i=7  date=JUL49 actual=148 season=6
i=8  date=AUG49 actual=148 season=7
i=9  date=SEP49 actual=136 season=8
i=10 date=OCT49 actual=119 season=9
i=11 date=NOV49 actual=104 season=10
i=12 date=DEC49 actual=118 season=11
i=13 date=JAN50 actual=115 season=0
i=14 date=FEB50 actual=126 season=1
i=15 date=MAR50 actual=141 season=2
i=16 date=APR50 actual=135 season=3
i=17 date=MAY50 actual=125 season=4
i=18 date=JUN50 actual=149 season=5
i=19 date=JUL50 actual=170 season=6
i=20 date=AUG50 actual=170 season=7
i=21 date=SEP50 actual=158 season=8
i=22 date=OCT50 actual=133 season=9
i=23 date=NOV50 actual=114 season=10
i=24 date=DEC50 actual=140 season=11
```

The seasonal cycle length is 12, and thus we see the zero-based seasonal index repeating. Though not used in this simple subroutine, all these data are available when you are computing forecasts.

Chapter 17
The HPFUCMSPEC Procedure

Chapter Contents

Chapter 17

The HPFUCMSPEC Procedure

Overview

The HPFUCMSPEC procedure is used to create a UCM model specification file. The output of this procedure is an XML file that stores the intended UCM model specification. This XML specification file can be used for different purposes; for example, it can be used to populate the model repository used by the HPFENGINE procedure (see Chapter 10, “The HPFENGINE Procedure”). You can specify any UCM model that can be analyzed using the UCM procedure; see Chapter 30, “The UCM Procedure” (*SAS/ETS User’s Guide*). Moreover, the model specification can include series transformations such as log or Box-Cox transformations. Apart from minor modifications to accommodate series transformations, the model specification syntax of the HPFUCMSPEC procedure is similar to that of the UCM procedure.

Getting Started

The following example shows how to create a UCM model specification file. In this example the specification for a Basic Structural Model (BSM) with one input is created.

```
proc hpfucmspec repository=sasuser.ucm
                name=BSM1
                label=
        "Basic structural model with one input";
   forecast symbol=Y transform=log;
   input symbol=X;
   irregular;
   level;
   slope variance=0 noest;
   season length=12 type=trig;
run;
```

The options in the PROC HPFUCMSPEC statement are used to specify the location of the specification file that will be output. Here the REPOSITORY= option specifies that the output file be placed in a catalog SASUSER.UCM, the NAME= option specifies that the name of the file be BSM1.xml, and the LABEL= option specifies a label for this catalog member. The other statements in the procedure specify the UCM model.

The model specification begins with the FORECAST statement that specifies a transformation, such as a log or Box-Cox, for the variable that is to be forecast. In some cases, the forecast variable is also called the *dependent* variable or the *response* variable. Here, the FORECAST statement specifies a log transformation for the series being forecast. The SYMBOL= option in the FORECAST statement can be used to provide a convenient name for the forecast variable. This name is only a placeholder, and a proper data variable will be associated with this name when this model specification is used in actual data analysis. Next, the INPUT statement specifies transformations, such as log or Box-Cox, as well as lagging and differencing, associated with the input variable. In this case the input variable enters the model as a simple regressor. Here again the SYMBOL= option can be used to supply a convenient name for the input variable. If a model contains multiple input variables then each input variable specification has to be given using a separate INPUT statement.

After the forecast and input series transformations are described, the components in the model are specified using different component statements. In the above example the model contains three components: an *irregular* component, a *local linear trend* with fixed *slope*, and a *trigonometric seasonal* with season length 12.

Syntax

The HPFUCMSPEC procedure uses the following statements.

PROC HPFUCMSPEC *options*;
  AUTOREG *options*;
  BLOCKSEASON *options*;
  CYCLE *options*;
  DEPLAG *options*;
  FORECAST *options* ;
  INPUT *options*;
  IRREGULAR *options*;
  LEVEL *options*;
  SEASON *options*;
  SLOPE *options*;

Functional Summary

The statements and options controlling the HPFUCMSPEC procedure are summarized in the following table.

Description	Statement	Option
Model Repository Options		
specify the model repository	PROC HPFUCMSPEC	REPOSITORY=
specify the model specification name	PROC HPFUCMSPEC	NAME=
specify the model specification label	PROC HPFUCMSPEC	LABEL=

Description	Statement	Option
Options for Specifying Symbolic Series Names		
specify a symbolic name for the response series	FORECAST	SYMBOL=
specify a symbolic name for the input series	INPUT	SYMBOL=
Options for Specifying the Model		
specify the response series transformation	FORECAST	TRANSFORM=
specify the input series transformation	INPUT	TRANSFORM=
specify the input series differencing orders	INPUT	DIF=
specify the input series lagging order	INPUT	DELAY=
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=

Description	Statement	Option
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=
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

PROC HPFUCMSPEC Statement

PROC HPFUCMSPEC *options ;*

The following options can be used in the PROC HPFUCMSPEC statement:

LABEL= *SAS-label*

specifies a descriptive label for the model specification to be stored in the SAS catalog or external file reference. The LABEL= option can also be specified as SPECLABEL=.

NAME= *SAS-name*

names the model specification to be stored in the SAS catalog or external file reference. The NAME= option can also be specified as SPECNAME=.

REPOSITORY= *SAS-catalog-name*
REPOSITORY= *SAS-file-reference*

names the SAS catalog or external file reference to contain the model specification. The REPOSITORY= option can also be specified as MODELREPOSITORY=, MODELREP=, or REP=.

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.

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.

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 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 series.

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 2 is an admissible value for the period, and the damping factor ρ can be any value in the interval (0, 1), including 1 but excluding 0. The cycles with the 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 1 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 that two cycles be included in the model. The parameters of each of these cycles is 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 are 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.

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.

DEPLAG Statement

DEPLAG LAGS= *order* <**PHI=** *value ...* > < **NOEST** > **;**

The DEPLAG statement is used to specify the lags of the forecast variable to be included as predictors in the model. The following examples illustrate the use of DEPLAG statement:

```
deplag lags=2;
```

If the forecast 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***)**
LAGS= **(***lag, ..., lag***)<**s_1**> ... (***lag, ..., lag***)<**s_k**>**

This is a required option in this statement. LAGS=(l_1, l_2, ..., l_k) defines a model with specified lags of the forecast 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.

You can also specify a multiplier after a parenthesized list. For example, LAGS=(1)(1)12 is equivalent to LAGS=(1)(12), and LAGS=(1,2)4(1)12(1,2)24 is equivalent to LAGS=(4,8)(12)(24,48).

NOEST
This option fixes the values of the parameters to those specified in PHI= options.

PHI= *value* **...**
lists starting values for the coefficients of the lagged forecast variable.

FORECAST Statement

FORECAST *options;*

The FORECAST statement specifies the symbolic name representing the series to be forecast as well as an optional transformation to be applied to the series. The symbolic name is used in later steps to associate actual time series variables with the model specification when the specification is applied to data.

The following options are used in the FORECAST statement.

(SYMBOL|VAR)= *variable*
specifies a symbolic name for the forecast series. This symbol specification is optional. If the SYMBOL= option is not specified, *Y* is used as a default symbol.

TRANSFORM= *option*
specifies the transformation to be applied to the time series. The following transformations are provided:

NONE	No transformation applied
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.

INPUT Statement

INPUT *options;*

The INPUT statements specify the inputs in the model. A separate INPUT statement is needed for each of the inputs. In this statement you can specify the delay order, the differencing orders, and the Box-Cox type transformations associated with the input variable under consideration. The following options are used in the INPUT statement.

DELAY= *order*
specifies the delay, or lag, order for the input series.

DIF= *order*
DIF= *(order1, order2, ...)*
specifies the differencing orders for the input series.

PREDEFINED= *option*
associates a predefined trend or a set of seasonal dummy variables with this transfer function. The SYMBOL= and PREDEFINED= options are mutually exclusive.

In the following list of options, let t represent the observation count from the start of the period of fit for the model, and let X_t be the value of the time trend variable at observation t.

LINEAR	A linear trend, with $X_t = t - c$
QUADRATIC	A quadratic trend, with $X_t = (t - c)^2$
CUBIC	A cubic trend, with $X_t = (t - c)^3$
INVERSE	An inverse trend, with $X_t = 1/t$
SEASONAL	Seasonal dummies. 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 a level component, and $X_{i,t} : 1 \leq i \leq (s), 1 \leq t \leq n$ for models that do not include a level component. Each element of a seasonal dummy regressor is either zero or one, based on the following rule:

$$X_{i,t} = \begin{cases} 1 & \text{when } i = t \\ 0 & \text{otherwise} \end{cases}$$

(SYMBOL|VAR)= *variable*
specifies a symbolic name for the input series. This symbol specification is optional. If the SYMBOL= option is not specified then *X* is used as a default symbol. If there are multiple INPUT statements then an attempt is made to generate a unique set of input symbols.

TRANSFORM= *option*
specifies the transformation to be applied to the time series. The following transformations are provided:

NONE	No transformation applied
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.

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 nonlinear parameter estimation process.

```
irregular variance=4;
```

NOEST
This option fixes the value of σ_ϵ^2 to the value specified in the VARIANCE= option.

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, forms 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 (see "SLOPE Statement"), 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. The value of σ_ξ^2, the disturbance variance in the slope equation, will be estimated from the data.

```
level variance=4 noest;
slope;
```

NOEST
This option fixes the value of σ_η^2 to the value specified in the VARIANCE= option.

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.

SEASON Statement

SEASON *< options >* **;**

The SEASON or the SEASONAL statement is used to specify a *seasonal* component, γ_t, in the model. A seasonal component 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 $\omega^*_{j,t} \sim N(0, \sigma^2_\omega)$. 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^2_ω, in the seasonal model.

There can be more than one seasonal component in the model, necessarily with different season lengths. Each seasonal component is specified using a separate SEASON statement. A model with multiple seasonal components can easily become quite complex and may need large amounts 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 4, corresponding to the quarterly seasonality. The disturbance variance σ^2_ω is estimated from the data. The following statement specifies a trigonometric seasonal with monthly seasonality. It also provides a starting value for σ^2_ω.

```
season length=12 type=trig variance=4;
```

LENGTH= *integer*

This option is used to specify the season length, *s*. The season length can be any integer larger than or equal to 2, or it can be "s", indicating a placeholder that will be substituted later with an appropriate value. The specification of season length is optional; in its absence it defaults to LENGTH=s. The use of specs with a placeholder for season lengths is further explained in Example 17.3. 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 disturbance variance parameter to the value specified in the VARIANCE= option.

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.

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 is ignored. The equations of the trend, defined jointly by the level μ_t and slope β_t, are as follows:

$$\begin{aligned}
\mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad & \eta_t &\sim i.i.d.\ N(0, \sigma_\eta^2) \\
\beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d.\ N(0, \sigma_\xi^2)
\end{aligned}$$

The SLOPE statement is used to specify the value of the disturbance variance, σ_ξ^2, in the slope equation, and to request forecasts of β_t. The following examples illustrate this statement:

```
level;
slope;
```

These statements request that a locally linear trend be used in the model. The disturbance variances σ_η^2 and σ_ξ^2 are 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_ξ^2, to the value specified in the VARIANCE= option.

VARIANCE= *value*

This option is used to supply an initial value for the disturbance variance, σ_ξ^2, in the β_t equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

Examples

Example 17.1. Some Syntax Illustrations

The following code fragments illustrate the HPFUCMSPEC syntax for some of the commonly needed modeling activities. Suppose that a variety of UCM models are to be fitted to a data set that contains a sales series as the forecast variable and several promotional events as predictor series. In all these cases the model repository is kept the same, sasuser.ucm, and the models are named as *model1, model2, ...* to ensure uniqueness. Note that in a given repository, the models must have unique names. The symbols for the forecast and input variables are *sales* and *promo1, promo2, ...*, respectively.

```
 /* BSM with two inputs */
 proc hpfucmspec repository=sasuser.ucm
                 name=model1;
    forecast symbol=sales transform=log;
    input symbol=promo1 delay=3;
    input symbol=promo2 dif=1;
    irregular;
    level;
    slope variance=0 noest; /* non-varying slope */
    season length=12 type=trig;
 run;
```

```
/* Model with one cycle and Box-Cox transform */
 proc hpfucmspec repository=sasuser.ucm
                 name=model2;
    forecast symbol=sales transform=BoxCox(0.8);
    irregular;
    level;
    slope;
    cycle rho=1 noest=(rho); /* fixed damping factor */
 run;
```

```
/* Unsaturated monthly seasonal */
 proc hpfucmspec repository=sasuser.ucm
                 name=model3;
    forecast symbol=sales transform=log;
    irregular;
    level;
    slope;
    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;
```

```
/* Supply starting values for the parameters */
 proc hpfucmspec repository=sasuser.ucm
                 name=model4;
    forecast symbol=sales transform=log;
    irregular;
    level;
    slope variance=10;
    cycle period=12 rho=0.9 noest=(period);
    cycle period=6 noest=(period);
 run;
```

Example 17.2. How to Include a UCM Model in a Model Selection List

One of the primary uses of the HPFUCMSPEC procedure is to add candidate UCM models to a model selection list that can be used by the HPFENGINE procedure (see Chapter 10, "The HPFENGINE Procedure"). The HPFUCMSPEC procedure is used to create the UCM model specifications and the HPFSELECT procedure is used to add the specifications to a model selection list (see Chapter 16, "The HPFSELECT Procedure"). This example illustrates this scenario.

Here a series that consists of the yearly river flow readings of the Nile, recorded at Aswan (see Cobb 1978), is studied. The data consists of readings from the years 1871 to 1970. This series is known to have had a shift in the level starting at the year 1899, and the years 1877 and 1913 are suspected to be outlying points.

The following DATA step statements read the data in a SAS data set and create dummy inputs for the shift in 1899 and the unusual years 1877 and 1913.

```
data nile;
   input riverFlow @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
   if year >= '1jan1899'd then Shift1899 = 1.0;
   else Shift1899 = 0;
   if year = '1jan1913'd then Event1913 = 1.0;
   else Event1913 = 0;
   if year = '1jan1877'd then Event1877 = 1.0;
   else Event1877 = 0;
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
;
```

Three candidate models are specified, *m1, m2*, and *m3*. Out of these three models *m1* is the simplest, which ignores the background information. Out of the other two models, *m2* uses only the shift in 1899, while *m3* uses all the three inputs. The following syntax shows how to specify these models and how to create a selection list that combines them using the HPFSELECT procedure. In the HPFSELECT procedure note the use of INPUTMAP option in the SPEC statement. It ties the symbolic variable names used in the HPFARIMASPEC procedure with the actual variable names in the data set. If the symbolic names were appropriate to start with, then the INPUTMAP option need not be used.

```
*make spec1;
proc hpfucmspec repository=sasuser.mycat
                name=m1;
   forecast symbol=y;
   irregular;
   level;
run;

*make spec2;
proc hpfucmspec repository=sasuser.mycat
                name=m2;
   forecast symbol=y;
   irregular;
   level;
   input symbol=x1;
run;

*make spec3;
proc hpfucmspec repository=sasuser.mycat
                name=m3;
   forecast symbol=y;
   irregular;
   level;
   input symbol=x1;
   input symbol=x2;
   input symbol=x3;
run;

*make a selection list that includes m1, m2 and m3;
proc hpfselect repository=sasuser.mycat
               name=myselect;

   spec m1 / inputmap(symbol=y var=riverFlow);

   spec m2 / inputmap(symbol=y var=riverFlow)
       inputmap(symbol=x1 var=Shift1899);

   spec m3 / inputmap(symbol=y var=riverFlow)
       inputmap(symbol=x1 var=Shift1899)
       inputmap(symbol=x2 var=Event1877)
       inputmap(symbol=x3 var=Event1913);
run;
```

This selection list can now be used in the HPFENGINE procedure for various types of analyses. The following syntax shows how to compare these models based on the default comparison criterion, Mean Absolute Percentage Error (MAPE). As expected, the model *m3* turns out to be the best of the three compared (see Output 17.2.1).

```
proc hpfengine data=nile
         repository=sasuser.mycat
         globalselection=myselect
         lead=0
         print=(select);

   forecast riverFlow;
   input    Shift1899;
   input    Event1877;
   input    Event1913;
run;
```

Output 17.2.1. Model Selection Based on the MAPE Criterion

```
Model           MAPE    Selected

 M1        13.096557      No
 M2        11.978729      No
 M3        10.532463      Yes
```

Example 17.3. How to Create a Generic Seasonal Model Spec That Is Suitable for Different Season Lengths

In the case of many seasonal model specifications, it is possible to describe a generic specification that is applicable in a variety of situations just by changing the season length specifications at appropriate places. As an example consider the Basic Structural model, which is very useful for modeling seasonal data. The Basic Structural model for a monthly series can be specified using the following syntax:

```
   proc hpfucmspec repository=work.specs
                   name=MonthlyBSM
                   label=
  "Basic Structural Model For A Series With Season Length 12";
      forecast symbol=Y transform=log;
      irregular;
      level;
      slope;
      season type=trig length=12;
   run;
```

It is easy to see that the same syntax is applicable to a quarterly series if the length in the SEASON specification is changed from 12 to 4. A generic specification that allows for late binding of season lengths can be generated by the following syntax:

```
proc hpfucmspec repository=work.specs
                name=GenericBSM
                label=
       "Generic Basic Structural Model";
   forecast symbol=Y transform= log;
   irregular;
   level;
   slope;
   season type=trig length=s;
run;
```

In this syntax the length in the SEASON specification is changed from 12 to “s”. This syntax creates a template for the Basic Structural model that is applicable to different season lengths. When the HPFENGINE procedure, which actually uses such model specifications to estimate the model and produce the forecasts, encounters such a “generic” specification it automatically creates a proper specification by replacing the season length placeholder with the value implied by the ID variable or its SEASONALITY= option. The following example illustrates the use of this generic spec. It shows how the same spec can be used for monthly and quarterly series. The parameter estimates for monthly and quarterly series are given in Output 17.3.1 and Output 17.3.2, respectively.

```
/* Create a selection list that contains
      the Generic Airline Model */
proc hpfselect repository=work.specs
               name=genselect;
   spec GenericBSM;
run;


/* Monthly interval */
proc hpfengine data=sashelp.air
               repository=work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=month;
   forecast air;
run;
```

Output 17.3.1. Parameter Estimates for the Monthly Series

```
                              The HPFENGINE Procedure

                                Parameter Estimates

                                                 Standard                 Approx
Component     Parameter          Estimate           Error    t Value    Pr > |t|

IRREGULAR     ERROR VARIANCE    0.0002344       0.0001079       2.17      0.0298
LEVEL         ERROR VARIANCE    0.0002983       0.0001057       2.82      0.0048
SLOPE         ERROR VARIANCE   9.8572E-13      6.7141E-10       0.00      0.9988
SEASON        ERROR VARIANCE   3.55769E-6      1.32347E-6       2.69      0.0072
```

```
/* Create a quarterly series illustrating accumulating
   the monthly Airline series to quarterly */
proc timeseries data=sashelp.air out=Qair;
   id date interval=quarter;
   var air / accumulate=total;
run;


/* Quarterly interval */
proc hpfengine data=Qair
               repository=work.specs
               globalselection=genselect
               print=(estimates);
   id date interval=quarter;
   forecast air;
run;
```

Output 17.3.2. Parameter Estimates for the Quarterly Series

```
                              The HPFENGINE Procedure

                                Parameter Estimates

                                                 Standard                 Approx
Component     Parameter          Estimate           Error    t Value    Pr > |t|

IRREGULAR     ERROR VARIANCE   6.7903E-11      1.32293E-7       0.00      0.9996
LEVEL         ERROR VARIANCE    0.0006273       0.0001762       3.56      0.0004
SLOPE         ERROR VARIANCE   1.1511E-11      1.68785E-8       0.00      0.9995
SEASON        ERROR VARIANCE   0.00002010      9.68319E-6       2.08      0.0379
```

References

Cobb, G. W. (1978), “The Problem of the Nile: Conditional Solution to a Change Point Problem,” *Biometrika,* 65, 243-251.

Part 3
Forecasting Details

Contents

Chapter 18

Forecasting Process Summary

Chapter Contents

Chapter 18
Forecasting Process Summary

Background

This chapter provides a brief theoretical background on automatic forecasting. An introductory discussion of automatic forecasting topics can be found in Makridakis, Wheelwright, and Hyndman (1997), Brockwell and Davis (1996), and Chatfield (2000). A more detailed discussion of time series analysis and forecasting can be found in Box, Jenkins, and Reinsel (1994), Hamilton (1994), Fuller (1995), and Harvey (1994).

This chapter also provides a summary of the SAS High-Performance Forecasting process. Forecasting steps, data and information flows, and information repositories are explained in this chapter.

Transactional Data

Transactional data are time-stamped data collected over time at no particular frequency. Some examples of transactional data are

- Internet data
- point-of-sale (POS) data
- inventory data
- call center data
- trading data

Businesses often want to analyze 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. The frequency and the season may vary with the business problem. Various statistics can be computed on each time period and season, for example:

- Web visits by hour and by hour of day
- sales per month and by month of year
- inventory draws per week and by week of month
- calls per day and by day of week
- trades per weekday and by weekday of week

Time Series Data

Time series data are time-stamped data collected over time at a particular frequency. Some examples of time series data are

- Web visits per hour
- sales per month
- inventory draws per week
- calls per day
- trades per weekday

As can be seen, the frequency associated with the time series varies with the problem at hand. The frequency or *time interval* may be hourly, daily, weekly, monthly, quarterly, yearly, or many other variants of the basic time intervals. The choice of frequency is an important modeling decision. This decision is especially true for automatic forecasting. For example, if you want to forecast the next four weeks, it is best to use weekly data rather than daily data. The forecast horizon in the former case is 4, while in the latter case it is 28.

Associated with each time series is a seasonal cycle or *seasonality*. For example, the length of seasonality for a monthly time series is usually assumed to be 12 because there are 12 months in a year. Likewise, the seasonality of a daily time series is usually assumed to be 7. The usual seasonality assumption may not always hold. For example, if a particular business' seasonal cycle is 14 days long, the seasonality is 14, not 7.

Time series that consist of mostly zero values (or a single value) are called interrupted or *intermittent time series*. These time series are mainly constant-valued except for relatively few occasions. Intermittent time series must be forecast differently from non-intermittent time series.

Forecasting Models

A skilled analyst can choose from a number of forecasting models. For automatic forecasting of large numbers of time series, only the most robust models should be used. The goal is not to have the analyst manually choose the very best model for forecasting each time series. The goal is to provide a list of *candidate models* that will forecast the large majority of the time series well. In general, when an analyst has a large number of time series to forecast, the analyst should use automatic forecasting for the low-valued forecasts; the analyst can then spend a larger portion of his time dealing with high-valued forecasts or low-valued forecasts that are problematic.

The candidate models that are used here are considered the most robust in the forecasting literature and these models have proven their effectiveness over time. These models consider the local level, local trend, and local seasonal components of the time series. The term *local* is used to describe the fact that these components evolve with time. For example, the local trend component may not be a straight line but a trend line whose slope changes with time. In each of these models, there is an error or random component that models the uncertainty.

The components associated with these models are not only useful for forecasting but also for describing how the time series evolves over time. The forecasting model decomposes the series into its various components. For example, the local trend component describes the trend (up or down) at each point in time, and the final trend component describes the expected future trend. These forecasting models can also indicate departures from previous behavior or can be used to cluster time series.

The parameter estimates (*weights* or *component variances*) describe how fast the component is changing with time. Weights or component variances near zero indicate a relative constant component; weights near one or large component variances indicate a relatively variable component. For example, a seasonal weight near zero or a component variance near zero represents a stable seasonal component; a seasonal weight near one or a large component variance represents an unstable seasonal component. Parameter estimates should be optimized for each time series for best results.

Local Level Models

The local level models are used to forecast time series whose level (or mean) component varies with time. These models predict the local level for future periods.

(Series) = (Local Level) + (Error)

Examples of local level models are Simple Exponential Smoothing and Local Level Unobserved Component Model. This model has one parameter (level), which describes how the local level evolves. The forecasts for the future periods are simply the final local level (a constant).

Local Trend Models

The local trend models are used to forecast time series whose level or trend/slope components vary with time. These models predict the local level and trend for future periods.

(Series) = (Local Level) + (Local Trend) + (Error)

Examples of local trend models are Double (Brown), Linear (Holt), Damped-Trend Exponential Smoothing, and Local Trend Unobserved Component Model. The double model has one parameter (level/trend weight), the linear model has two parameters (level and trend), and the damped-trend model has three parameters (level, trend, and damping weights). The damping weight dampens the trend over time. The forecasts for the future periods are a combination of the final local level and the final local trend.

Local Seasonal Models

The local seasonal models are used to forecast time series whose level or seasonal components vary with time. These models predict the local level and season for future periods.

(Series) = (Local Level) + (Local Season) + (Error)

Examples of local seasonal models are Seasonal Exponential Smoothing and the Local Seasonal Unobserved Component Model. The seasonal model has two parameters (level and seasonal). The forecasts for the future periods are a combination of the final local level and the final local season.

Local Models

The local models are used to forecast time series whose level, trend, or seasonal components vary with time. These models predict the local level, trend, and seasonal component for future periods.

(Series) = (Local Level) + (Local Trend) + (Local Season) + (Error)

(Series) = ((Local Level) + (Local Trend)) x (Local Season) + (Error)

Examples of local models are the Winters Method (additive or multiplicative) and the Basic Structural Model. These models have three parameters (level, trend, and seasonal). The forecasts for the future periods are a combination of the final local level, the final local trend, and final local season.

ARIMA Models

The Autoregressive Integrated Moving Average Models (ARIMA) are used to forecast time series whose level, trend, or seasonal properties vary with time. These models predict the future values of the time series by applying non-seasonal or seasonal polynomial filters to the disturbances. Using different types of polynomial filters permits the modeling of various properties of the time series.

(Series) = DisturbanceFilter (Error)

Examples of ARIMA models are the Exponentially Weighted Moving Average (EWMA), moving average processes (MA), integrated moving average processes (IMA), autoregressive processes (AR), integrated autoregressive processes (IAR), and autoregressive moving average processes (ARMA).

Causal Models

Causal time series models are used to forecast time series data that are influenced by causal factors. Input variables (regressor or predictor variables) and calendar events (indicator, dummy, or intervention variables) are examples of causal factors. These independent (exogenous) time series causally influence the dependent (response, endogenous) time series and, therefore, can aid the forecasting of the dependent time series.

Examples of causal time series models are Autoregressive Integrated Moving Average with eXogenous inputs (ARIMAX), which are also known as transfer function models or dynamic regression models, and Unobserved Component Models (UCM), which are also known as state-space models and structural time series models.

(Series) = TransferFunctionFilter(Causal Factors) + DisturbanceFilter(Error)

(Series) = (Local Level) + (Local Trend) + (Local Season) + (Causal Factors) + (Error)

These regression models are *dynamic* because they take into account the autocorrelation between observations recorded at different times. Dynamic regression includes and extends multiple linear regression (static regression).

Input variables are typically continuous-valued time series. They represent causal factors that influence the dependent time series throughout the time range. Examples of input variables are prices, temperatures, and other economic or natural factors. Input variables are contained in the time series data set.

Calendar events can be represented by indicator variables that are typically discrete-valued. They indicate when the causal factor influences the dependent time series. Typically, zero values indicate the absence of the event and nonzero values indicate the presence of the event. These dummy regressors can consist of pulses (points), steps (shifts), ramps, and temporary changes and combinations of these primitive shapes. The values of the indicator variable depend on the time interval. For example, if the calendar event is New Year's Day and the time interval is monthly, a pulse indicator variable will be nonzero for each January and zero otherwise.

In addition to the causal factors, the causal model can contain components described in preceding sections: local level, local trend, and local seasonal. Causal models decompose the time series into causal factors and the local components. This decomposition is useful for demand analysis (promotional analysis and intervention analysis).

Transformed Models

With the exception of the Winters Method Multiplicative Model, the preceding forecasting models are linear; that is, the components must be added together to re-create the series. Since time series are not always linear with respect to these components, transformed versions of the preceding forecasting models must be considered when using automatic forecasting. Some useful time series transformations are

- Logarithmic
- Square-Root
- Logistic
- Box-Cox

For example, suppose the underlying process that generated the series has one of the following nonlinear forms:

(Series) = Exp ((Local Level) + (Local Trend) + (Error)) exponential growth model

(Series) = (Local Level) x (Local Season) x (Error) multiplicative error model

Transforming the preceding series permits the use of a linear forecasting model:

Log(Series) = (Local Level) + (Local Trend) + (Error) log local trend model

Log(Series) = Log(Local Level) + Log(Local Seasonal) + Log(Error) log local seasonal model

The preceding transformations can only be applied to positive-valued time series.

Intermittent Demand Models

Intermittent demand models (IDM) or interrupted time series models are used to forecast intermittent time series data. Since intermittent series are mostly constant valued (usually zero) except on relatively few occasions, it is often easier to predict when the series departs and how much the series departs from this constant value rather than the next value. An example of an intermittent demand model is Croston's Method.

Intermittent demand models decompose the time series into two parts: the interval series and the size series. The interval series measures the number of time periods between departures. The size series measures the magnitude of the departures. After this decomposition, each part is modeled and forecast independently. The interval forecast predicts when the next departure will occur. The size forecast predicts the magnitude of the next departure. After the interval and size predictions are computed, they are combined (predicted magnitude divided by predicted number of periods for the next departure) to produce a forecast for the average departure from the constant value for the next time period.

External and User-Defined Models

In addition to the previously described general classes of Exponential Smoothing Models (ESM), Unobserved Component Models (UCM), Autoregressive Integrated Moving Average Models (ARIMA), and Intermittent Demand Models (IDM), HPF allows for external models and user-defined models.

External models are used for forecasts that are provided external to the system. These external forecasts may have originated from an external statistical model from another software package, may have been provided by an outside organization (e.g., marketing organization, government agency) or may be based on judgment. External models allow for the evaluation of external forecasts and for tests for unbiasedness.

User-defined models are external models that are implemented with the SAS programming language or the C programming language by the user of HPF software. For these models, users of HPF create their own computational algorithm to generate the forecasts. They are considered external models because they were not implemented in HPF.

Forecasts

Forecasts are time series predictions made for future periods. They are random variables and, therefore, have an associated probability distribution. For example, assuming a normal distribution, the forecasts for the next three months can be viewed as three "bell-curves" that are progressively flatter (or wider). The mean or median of each forecast is called the *prediction*. The variance of each forecast is called the *prediction error variance* and the square root of the variance is called the *prediction standard error*. The variance is computed from the forecast model parameter estimates and the model residual variance.

The forecast for the next future period is called the *one-step-ahead forecast*. The forecast for *h* periods in the future is called the *h-step-ahead forecast*. The *forecast horizon* or *forecast lead* is the number of periods into the future for which predictions are made (one-step, two-step,..., *h*-step). The larger the forecast horizon, the larger the prediction error variance at the end of the horizon. For example, forecasting daily data four weeks into the future implies a forecast horizon of 28, whereas forecasting weekly data four weeks into the future implies a forecast horizon of only 4. The prediction standard error at the end of the horizon in the former case may be larger than the prediction standard error in the latter case.

The *confidence limits* are based on the prediction standard errors and a chosen confidence limit size. A confidence limit size of 0.05 results in 95% confidence limits. The confidence limits are often computed assuming a normal distribution, but others could be used. As with the prediction standard errors, the width of the confidence limits increases with the forecast horizon. Once again, the forecast horizon of 28 will have wide confidence limits at the end of the horizon, representing greater uncertainty.

The *prediction error* is the difference between the actual value and the predicted value when the actual value is known. The prediction errors are used to calculate the statistics of fit that are describe later. For transformed models, it is important to understand the difference between the model errors (or residuals) and the prediction errors. The residuals measure the departure from the model in the transformed metric (Log, Square Root, etc.). The prediction errors measure the departure from the original series. You should not directly compare the model residuals of a transformed model and a non-transformed model when evaluating the model fit. You can compare the prediction errors between any two models because prediction errors are computed on the same metric.

Taken together, the predictions, prediction standard errors, and confidence limits at each period in the forecast horizon are the *forecasts*. Although many people use the term "forecast" to imply only prediction, a forecast is not one number for each future time period.

Using a transformed forecasting model requires the following steps:

- The time series data are transformed.
- The transformed time series data are fit using the forecasting model.
- The forecasts are computed using the parameter estimates and the transformed time series data.
- The forecasts (predictions, prediction standard errors, and confidence limits) are inverse transformed.

The naive inverse transformation results in *median forecasts*. To obtain *mean forecasts* requires that the prediction and the prediction error variance both be adjusted based on the transformation. Additionally, the model residuals will be different from the prediction errors due to this inverse transformation. If no transformation is used, the model residual and the prediction error will be the same, and likewise the mean and median forecast will be the same (assuming a symmetric disturbance distribution).

For causal models, the future values of the causal factors must be provided in order to forecast the time series. A causal factor is *deterministic* if its future values are known with certainty. A causal factor is *controllable* if its future values are under the control of the organization producing the forecasts. A causal factor is *stochastic* if its future values are not known with certainty. If the causal factor is *stochastic*, it must be forecast as well, and the uncertainty of its forecast (prediction standard errors) must be incorporated into the uncertainty of the time series forecast.

Forecast Function (Scoring)

For causal models that include controllable causal factors, the predictions can be influenced by the future decisions made by the organization producing the forecasts. Changing the future values of the controllable causal factors changes the forecasts. Organizations want to make decisions that benefit themselves. To help organizations make better decisions, the future values of the controllable causal factors can be varied to their benefit. The future values of the causal factors can be varied for scenario analysis (What-If analysis), stochastic optimization, or goal seeking to aid proper decision-making.

In scenario analysis, the organization sets the future values of the causal factors to specific values and then evaluates the effect on the forecasts. In stochastic optimization, the organization algorithmically varies the future values of the causal factors to find the optimum of an objective function (profit, revenue, or cost function) based on the forecasts. In goal seeking, the organization algorithmically varies the future values of the causal factors in order to determine the values that achieve a certain goal (profit, revenue, or cost goal) based on the forecasts.

For example, suppose the following:

- An organization desires to predict the demand for a product or service.
- The demand is influenced by its sales price and by its advertising expenditures.
- These data are recorded over time.

The following types of analysis may be used to answer questions about the time series data:

- Scenario analysis can help answer the question *What happens to demand if the organization increases the sales price and decreases the advertising expenditures?*
- Stochastic optimization can help answer the question *What is the optimal sales price and advertising expenditure combination that maximizes profit?*
- Goal seeking can help answer the question *What are the combinations of sales price and advertising expenditures that achieve a specified sales target?*

The sales price and advertising expenditures for a given time period may influence demand in future time periods. Static regression ignores these dynamic effects, which often leads to poor predictions, which in turn leads to poor decisions. Dynamic regression captures these dynamic effects and provides better predictions, which in turn facilitates better decisions.

Forecast score files (or forecast functions) summarize the time series model's parameter estimates and the final states (historical time series information). These files can be used to quickly generate the forecasts required for the iterative nature of scenario analysis, stochastic optimization, and goal-seeking computations. Since most of the computational effort associated with automatic forecasting is time series analysis, diagnostics, model selection, and parameter estimation, forecast scoring is relatively effortless. Therefore, forecast scoring makes the iterative nature of large scale decision-making more tractable.

The results of forecast scoring include the predictions, prediction standard errors, and the confidence limits. All of these results can be used in decision-making.

Statistics of Fit

The *statistics of fit* evaluate how well a forecasting model performs by comparing the actual data to the predictions. For a given forecast model that has been fitted to the time series data, the model should be checked or evaluated to see how well it fits or forecasts the data. Commonly used statistics of fit are Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Akaike Information Criteria (AIC), and many others. The statistics of fit can be computed from the model residuals or the prediction errors.

When the full range of data is used to both fit and evaluate the model, this is referred to as *in-sample evaluation*. When the most recent data are excluded for parameter estimation (holdout) and this *holdout sample* is used for evaluation, this is referred to as *holdout sample evaluation*. Holdout sample analysis is similar to *training* and *testing* of neural networks. A portion of the data is withheld from training (fit) and the withheld data (holdout) are used to test performance.

When a particular statistic of fit is used for forecast model selection, it is referred to as the *model selection criterion*. For example, if the MAPE (an often recommended choice) is used as a model selection criterion, the forecast model with smallest MAPE in the evaluation region (in-sample or holdout-sample) is chosen as the *best model*.

When a particular statistic of fit is used to judge how well the forecasting process is predicting the future, it is referred to as the *performance statistic*.

Automatic Forecasting Process

Automatic forecasting is usually defined as forecasting without the aid of an analyst skilled in time series analysis techniques or as forecasting when the number of forecasts is too numerous for an analyst to investigate. Automatic forecasting is usually performed on each time series independently. For each time series and for each candidate model, the parameter estimates are optimized for best results. This means that several optimizations may be required for each time series.

Accumulation Step

The *accumulation* of time-stamped data into time series data is based on a particular frequency. For example, time-stamped data can be accumulated to form hourly, daily, weekly, monthly, or yearly time series. Additionally, the method for accumulating the transactions within each time period is based on a particular statistic. For example, the sum, mean, median, minimum, maximum, standard deviation, and other statistics can be used to accumulate the transactions within a particular time period.

For automatic forecasting, accumulation is the most important decision because the software makes most of the remaining decisions. If weekly forecasts of the average of the transactions are needed, then the accumulation frequency should be weekly and the accumulation statistic should be the average.

Accumulating the transactional data on a relatively small time interval may require a long forecast horizon. For example, if the data are accumulated on an hourly basis and if it is desired to forecast one month into the future, the forecast horizon is very long and the width of the confidence limits will be very wide toward the end of the horizon. In this situation, the forecast content or usefulness of the forecast will be low.

Interpretation Step

Once the time-stamped data has been accumulated, there may be no data recorded for certain time periods (resulting in missing values in the accumulated time series). These missing values can represent unknown values (and so they should remain missing) or they can represent no activity (in which case they should be set to zero or some other appropriate value). Some transactional databases set missing data at the beginning or end of the time series to zero values. These zero values should be set to missing values. Missing values and zero values need to be interpreted before analyzing the time series.

Adjustment Step

Once the time-stamped data has been accumulated and interpreted, the time series to forecast may require adjustment prior to analysis or *pre-forecast adjustment*. By adjusting the time series for known systematic variations or deterministic components, the underlying stochastic (unknown) time series process may be more readily identified and modeled.

Examples of systematic adjustments are currency-unit conversions, exchange rates, trading days, and other known systematic variations. Examples of deterministic adjustments are advanced bookings and reservations, contractual agreements, and other known contributions or deterministic components.

After analysis, the statistical forecast of the adjusted time series may require *post-forecast adjustment* to return forecasts in the original metric.

Typically the pre-forecast and post-forecast adjustments are operations that are inverses of each other. For example, to adjust a time series for exchange rates, it is often desirable to

1. *Divide* the time series by the exchange rate.
2. Analyze and forecast the adjusted time series without regard to exchange rates.
3. Adjust the forecasts, *multiplying* by the exchange rate.

(Division and multiplication are inverse operations of each other.)

For another example, to adjust a time series for advanced bookings, it is often desirable to

1. *Subtract* the advanced bookings from the time series.
2. Analyze and forecast the adjusted time series without regard to advanced booking.
3. Adjust the forecasts, *adding* the advanced bookings.

(Subtraction and addition are inverse operations of each other.)

Systematic variations or deterministic components are included in the time series data. Adjustments are data whose effect is *excluded* prior to statistical analysis. Causal factors are data whose effect is *included* with the statistical analysis.

Diagnostic Step

Given the time series data, the time series diagnostics subset the potential list of candidate models to those that are judged appropriate to a particular time series. Time series that have trends (deterministic or stochastic) should be forecast with models that have a trend component. Time series with seasonal trends (deterministic or stochastic) should be forecast with models that have a seasonal component. Time series that are nonlinear should be transformed for use with linear models. Time series that are intermittent should be forecast with intermittent models.

The importance of the diagnostics should not be underestimated. Applying a seasonal model to a nonseasonal time series, particularly one with a short history, can lead to over parameterization or false seasonality. Applying a linear model to a nonlinear time series can lead to underestimation of the growth (or decline). Applying a non-intermittent model to an intermittent series will result in predictions biased toward zero.

If it is known, *a priori*, that a time series has a particular characteristic, then the diagnostics should be overridden and the appropriate model should be used. For example, if the time series is known to be seasonal, the diagnostics should be overridden to always choose a seasonal model.

There may be several causal factors that may or may not influence the dependent time series. The multivariate time series diagnostics determine which of the causal factors *significantly* influence the dependent time series. These diagnostics include cross-correlation analysis and transfer function analysis.

Once again, if it is known, *a priori*, that a particular causal factor is known to influence the dependent time series, then the diagnostics should be overridden and the appropriate model should be used.

Model Selection Step

After the candidate models have been subset by the diagnostics, each model is fit to the data (with the holdout sample excluded). After model fitting, the one-step-ahead forecasts are made in the fit region (in-sample) or the multistep-ahead forecasts are made in the holdout sample region (out-of-sample). The model selection criterion is used to select the best performing model from the appropriate subset of the candidate models. As described above, the model selection criteria are statistics of fit.

If the length of the time series is short, holdout sample analysis may not be possible due to a lack of data. In this situation, the full-range of the data should be used for fitting and evaluation. Otherwise, holdout sample analysis is recommended.

Parameter Estimation Step

Once the best forecasting model is selected from the candidate models, the selected model is fit to the full range of the data to obtain the most accurate model parameter estimates. If you excluded the holdout sample in this step, you would be ignoring the most recent and influential observations. Most univariate forecasting models are weighted averages of the past data, with the most recent having the greatest weight. Once the model is selected, excluding the holdout sample can result in poor forecasts. Holdout sample analysis is only used for forecast model selection, not for forecasting.

Forecasting Step

Once the model parameters are estimated, forecasts (predictions, prediction standard errors, prediction errors, and confidence limits) are made using the model parameter estimates, the model residual variance, and the full-range of data. If a model transformation was used, the forecasts are inverse transformed on a mean or median basis.

When it comes to decision-making based on the forecasts, the analyst must decide whether to base the decision on the predictions, lower confidence limits, upper confidence limits or the distribution (predictions and prediction standard errors). If there is a greater penalty for over predicting, the lower confidence limit should be used. If there is a greater penalty for under predicting, the upper confidence limit should be used. Often for inventory control decisions, the distribution (mean and variance) is important.

Evaluation Step

Once the forecasts are made, the in-sample statistics of fit are computed based on the one-step-ahead forecasts and the actual data. These statistics can be used to identify poorly fitting models prior to making business decisions based on these forecasts. If forecasts do not predict the actual data well, they can be flagged to signal the need for more detailed investigation by the analyst.

In addition to the statistics of fit, distribution and correlation analysis of the prediction errors can help evaluate the adequacy of the forecasting model.

Performance Step

The previous steps are used to forecast the future. This ex-post forecast evaluation judges the performance of the forecasting model. After forecasting future periods, the actual data becomes available as time passes. For example, suppose that monthly forecasts are computed for the next three months into the future. After three months pass, the actual data are available. The forecasts made three months ago can now be compared to the actual data of the last three months.

The availability of the new data begs the following questions:

- How well are you forecasting?
- Why are you forecasting poorly?
- If you were forecasting well before, what went wrong?

Some useful measures of forecast performance are the statistics of fit described in a preceding section. When the statistics of fit are used for performance measures, the statistics are computed from the previous predictions and the newly available actual data in the forecast horizon. For example, the MAPE can be computed from the previous predictions and the newly available actual data in the three-month forecast horizon.

Another useful measure of forecast performance is determining whether the newly available data fall within the previous forecasts' confidence limits. For example, performance could be measured by whether or not the newly available actual data fall outside the previous forecasts' confidence limits in the three-month forecast horizon.

If the forecasts were judged to be accurate in the past, a poor performance measure, such as actual data outside the confidence limits, could also be indicative of a change in the underlying process. A change in behavior, an unusual event, or other departure from past patterns may have occurred since the forecasts were made.

Such departures from past trends may be normal, and indicate the need to update the forecasting model selection for this variable, or they can be a warning of special circumstances that warrant further investigation.

Large departures from forecast can sometimes reflect data errors, changes in policies or data definitions (for example, what exactly is counted as sales), fraud, or a structural change in the market environment.

Forecast Function (Score File) Generation Step

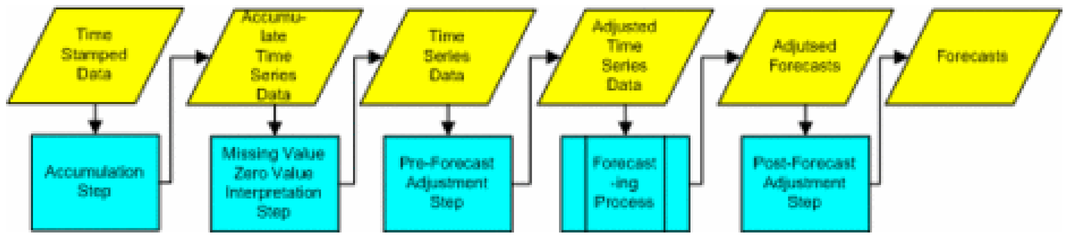
Once the selected model is fit to the full range of the data, a summary of model parameter estimates and the final states (historical time series information) are stored in a forecast score file. Subsequent decision-making processes can the use the forecast score file for scenario (What-If) analysis, stochastic optimization, or goal seeking.

Automatic Forecasting Data

For forecast scoring, the future values of the controllable causal factors must be specified by the user (scenario analysis) or iteratively generated by the decision process (stochastic optimization or goal seeking).

Automatic Forecasting Data Flow

The input and output of the automatic forecasting process is the time-stamped data set and the forecasts, respectively. The following diagram describes the automatic forecasting data flow.



Time Stamped Data
Accumulation Step
Accumu-late Time Series Data
Missing Value Zero Value Interpretation Step
Time Series Data
Pre-Forecast Adjustment Step
Adjusted Time Series Data
Forecast -ing Process
Adjutsed Forecasts
Post-Forecast Adjustment Step
Forecasts

Figure 18.1. Automatic Forecasting Data Flow

Forecast Scoring Data Flow

The input and output of the forecast scoring process are the future values of the controllable causal factors and the forecasts, respectively. The following diagram illustrates the forecast scoring data flow.

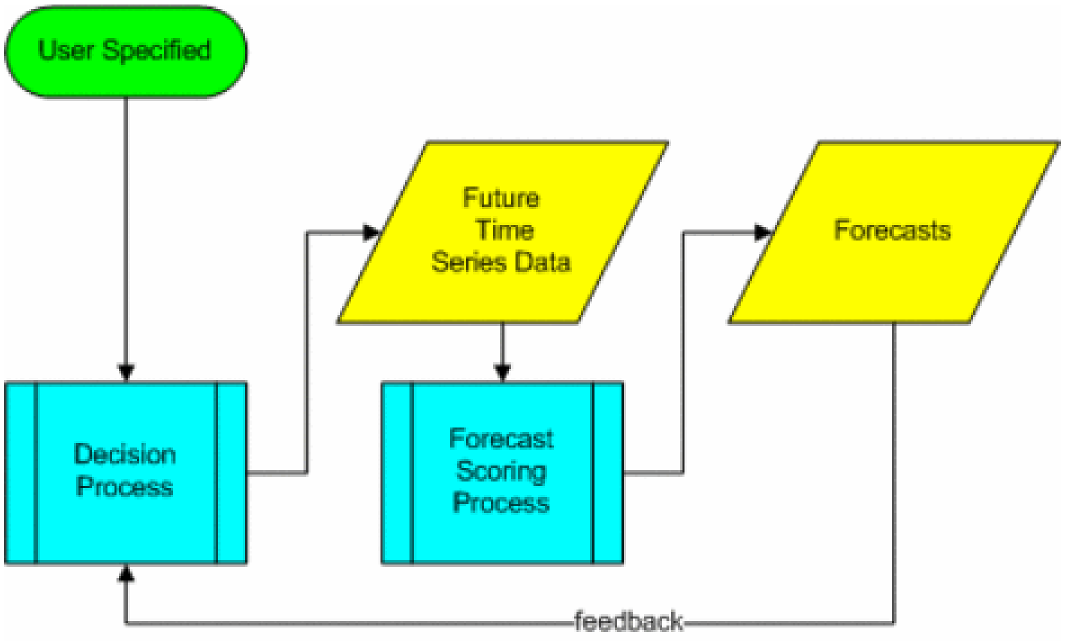


Figure 18.2. Forecast Scoring Data Flow

Automatic Forecasting Information

To automatically forecast a single time series, the time series must be diagnosed, selected, or specified to obtain a selected model used for forecasting. These abstract statistical concepts must be made concrete and persisted in a computer's storage. In addition to the time series data, the following information is needed.

Model Specification

A *model specification* indicates that a specific type of forecasting model be fit to the historical data and used to produce forecasts. Given a time series and a model specification, a forecast for the time series is generated by applying the abstract statistical concepts associated with model specification. A model specification is not dependent on any specific time series data; a given specification can be used for many different series.

Associated with a model specification is a list of symbols representing the time series to which the specification applies. These symbols must be mapped to actual time series variables in the input data set, or to event specifications, before the model specification can be used to created a fitted model for forecasting.

The following theoretical time series models are supported: ESM, IDM, ARIMAX, UCM, EXTERNAL, USER-DEFINED.

Except for the External and User-Defined models, all of the models are implemented to allow nonlinear transformations (Log, Square Root, Logistic, Box-Cox) of the dependent and independent time series.

Exponential Smoothing Models (PROC HPFESMSPEC)

Exponential smoothing models are extrapolation methods that predict future values based on exponentially weighted past values of the time series.

The following exponential smoothing models are supported:

- Simple Exponential Smoothing (SIMPLE)
- Double Exponential Smoothing (DOUBLE)
- Linear Exponential Smoothing (LINEAR)
- Damped-Trend Exponential Smoothing (DAMPTREND)
- Seasonal Exponential Smoothing (SEASONAL)
- Multiplicative Winters Method (WINTERS)
- Additive Winters Method (ADDWINTERS)

Intermittent Demand Models (PROC HPFIDMSPEC)

Intermittent Demand Models are extrapolation methods that predict future values based on exponentially weighted past values of intermittent (interrupted) time series components. These methods use nonseasonal exponential smoothing models to forecast the intermittent time series components (interval, size, average demand), independently.

The following intermittent demand models are supported:

- Croston's Method (CROSTON)
- Average Demand (AVERAGE)

Autoregressive Moving Average with Exogenous Inputs (HPFARIMASPEC)

ARIMAX models implement Box-Jenkins models with or without transfer function inputs.

The following ARIMA models are supported:

- Simple and Seasonal ARIMA
- Factored and Subset ARIMA
- Preceding models with Simple and Seasonal Transfer Function Inputs
- Preceding models with Factored and Subset Transfer Function Inputs

Unobserved Component Models with Exogenous Inputs (HPFUCMSPEC)

UCM models implement structural time series models with or without input variables.

The following UCM models are supported:

- Local Level
- Local Slope (or Trend)
- Local Seasons (up to three seasons)
- Local Cycles (no limit to the number of cycles)
- Exogenous Inputs
- Combinations of the preceding components

External Models (HPFEXMSPEC)

EXTERNAL models are forecasts provided by methods external to the system. These methods may be judgmental inputs or forecasts provided by an external system such as another forecasting system. These forecasts must be recorded in the input time series data set.

When only the future predictions are provided, prediction standard errors and confidence limits are computed using the past prediction errors, if available. These additional forecasting components can be computed assuming nonlinear transformations (Log, Square Root, Logistic, Box-Cox) and autocorrelation (White Noise, Prediction Error Autocorrelation, Series Autocorrelation).

Since the system has no knowledge of how the forecasts were computed, there are no parameter estimates. However, the forecast bias can be computed and a test for unbiasedness can be made.

User-Defined Models (USERDEFINED)

USERDEFINED models are forecasting methods provided by the user of the system. These methods are implemented in the SAS language or the C language. Since the system has no knowledge of how the forecasts were computed, the forecasts are treated as if they were EXTERNAL forecasts.

There is usually more than one model specification associated with a model repository. A model specification does not depend on a particular time series and a particular model specification can be assigned to different time series. However, a unique model specification must be assigned or selected for each time series in order to forecast the time series.

A model specification can also be referenced in one or more model selection lists.

The model specification is stored in an XML format and this format follows the spirit of the PMML specification. It is stored as a SAS catalog entry or as an external file.

See Chapter 21, “User-Defined Models,” for more information.

Model Selection List

A *model selection list* specifies a list of candidate model specifications and how to choose which model specification is best suited to forecast a particular time series. Given a time series and an appropriate model selection list, a forecasting model can be automatically selected for the time series. Since the model selection process is applied to each series individually, the process may select a different model for different series and may select a different model for a given time series, with the passage of time as more data are collected. A model selection list is not dependent on any specific time series data.

A model selection list consists of the following:

List of Candidate Model Specifications	Specifies the list of model specifications to consider when choosing the model for forecasting.
Selection Diagnostics	Specifies how to subset the list of model specifications models to those that are judged appropriate to a particular time series.
Holdout Sample Size	Specifies the size of the holdout sample region. A holdout sample size of zero indicates that the full range of data is used to both fit and evaluate the forecast. The holdout sample size can be an absolute size or a percentage of the length of the time series data.
Model Selection Criterion	Specifies the statistic of fit to be used to select the best performing model from the subset list of the candidate models returned by the selection diagnostics.
Confidence Limit Size	Specifies the confidence limit size for computing lower and upper confidence limits.

There may be more than one model selection list associated with a model repository. A model selection list does not depend on a particular time series, and a particular model selection list can be assigned to different time series. However, a unique model selection list must be assigned to each time series. If desired, each time series to be forecast can have its own model selection list; typically, for time series with similar characteristics, the same model selection list is assigned.

The model selection list is stored in an XML format and this format follows the spirit of the PMML specification. It is stored as a SAS catalog entry or as an external file.

See Chapter 16, "The HPFSELECT Procedure," for more information.

Selected Model Specification

A *selected model specification* is a model specification that is the result of the diagnostic or model selection processes for a particular time series. The selected model specification is used to forecast this time series.

The file reference of the selected model specification is stored in a SAS data set.

Fitted Model

A *fitted model* results from applying a model specification to specific time series data. Given a time series and a (diagnosed, selected, or specified) model specification, the model parameter estimates can be optimized to fit the time series data. The fitted model is used to forecast this time series.

The parameter estimates associated with fitted models are stored in a SAS data set.

Forecast Function (Score File)

A *forecast model score file* encodes the information needed to compute forecasts for a time series given the future values of the causal factors. Given a time series and a (diagnosed, selected, or specified) model specification, a fitted time series model is estimated. Given a fitted model and a time series, a forecast model score file can be generated that efficiently encapsulates all information needed to forecast the series when future inputs are provided.

The forecast model score file is stored in an XML format that follows the spirit of the PMML score file. It is stored as a SAS catalog entry or as an external file.

Forecast model score files can be used for scenario analysis, goal seeking, or stochastic optimization. SAS functions are provided that can reference the forecast model score files to calculate forecasts from the fitted model given alternative inputs. These functions can be used in user-written SAS Data Set programs or in SAS analytical procedures such as PROC MODEL or PROC NLP.

See Chapter 20, “Using Scores,” for examples and additional information.

Automatic Forecasting Information Flow

SAS HPF is designed to support fully automated forecasting. HPF also allows you a great deal of control over the forecasting process when you which to override steps in the automatic process. You can control any or all of the forecasting steps, or allow the system to control all steps.

The degree of automation depends on how much information you specify. For each time series, the information flow of the automatic forecasting technique is described in the following diagram:

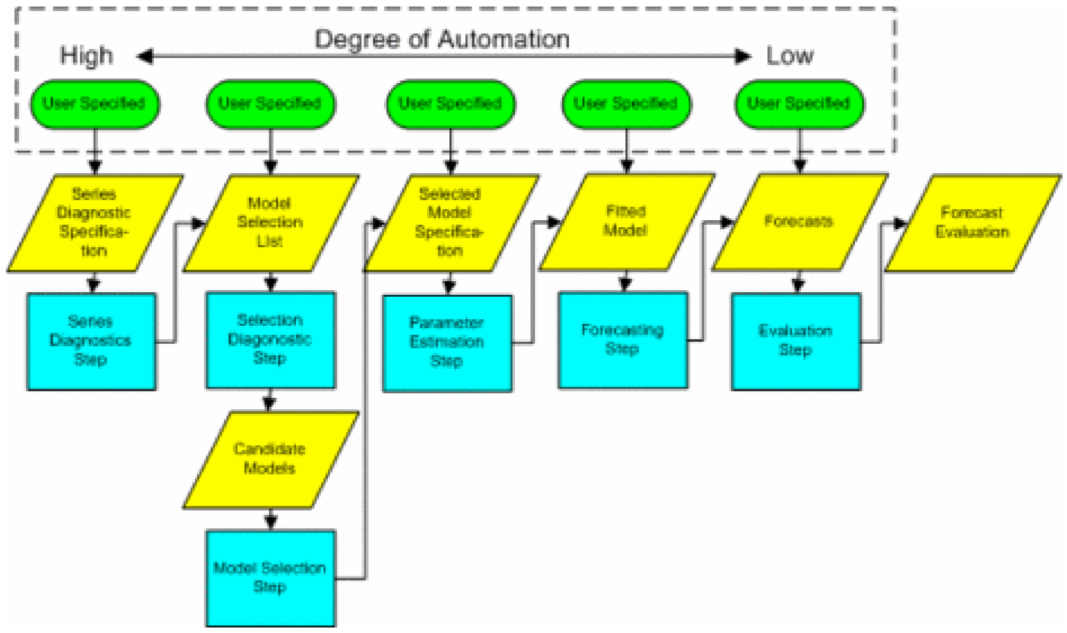


Figure 18.3. Automatic Forecasting Information Flow

The more information provided by the user, the less automation is needed. If the user specifies the forecasts (external forecasts), nothing is required. If the user specifies the fitted model, only forecasting is required. If the user specifies the selected model specification, then parameter estimation and forecasting are required. If the user specifies a model selection list, then model selection, parameter estimation, and forecasting are required. If the diagnostic specification is specified, then diagnostics, model selection, parameter estimation, and forecasting are required. If the user specifies nothing, the default diagnostics or model selection list is used.

The more information provided by the user, the less computational effort is needed. Series diagnostics are the most expensive, followed by model selection, parameter estimation, and forecasting. Since the information is persisted in the computer's storage, differing degrees of automation can be used over time. For instance, it may be desirable to use the diagnostic step every six months, the model selection step every three, the parameter estimation step every month, and the forecasting step every week. This staggering of the degree of automation reduces the processing time by allowing the most up-to-date information about the time series data to influence the automatic forecasting process over time.

Forecast Scoring Information Flow

For each time series, the information flow of the forecast scoring technique presented here is described in the following diagram:

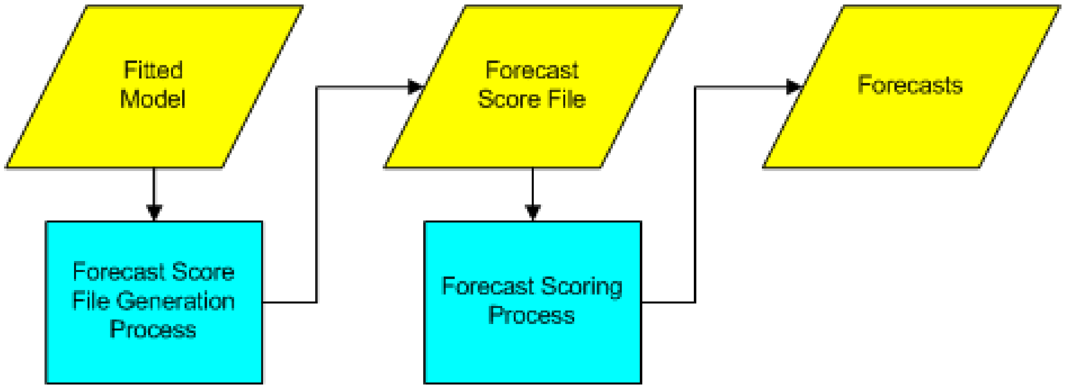


Figure 18.4. Forecast Scoring Information Flow

A fitted model generates a forecast score file. Using the forecast score file and given the future values of the controllable causal factors, the forecast scoring process generates the forecasts.

Automatic Forecasting Repositories

Since there are many time series to forecast, large-scale automatic forecasting requires the efficient management of large amounts of information about each time series. In addition to the time series data, the following information repositories are needed.

Event Repository

An *event repository* stores information about calendar events using a brief description of each event. Calendar events can be represented by indicator variables that could be stored in the time series data. However, because the influential calendar events can vary from series to series, there may be too many to store efficiently and many calendar events will be redundant, making updates difficult. Therefore, it is better to store a brief description of the calendar event, to reproduce the indicator variable in the computer's memory when needed, and to store the calendar events independently of the time series data, to allow the reuse and update of the calendar events. Additionally, the event repository can be used by more than one time-stamped data set.

See Chapter 12, "The HPFEVENTS Procedure," for more information about creating event definitions and storing them in an event repository.

Model Specification Repository

A *model specification repository* stores information about time series models (model specification) and how to select an appropriate time series model (model selection list) when given a particular time series. A model specification can be assigned to each time series. However, because the model specification can vary from series to series, there may be too many to store efficiently and many model specifications will be redundant, making updates difficult. Therefore, it is better to store model specifications independently of the time series data to allow the reuse and update of the model specification. Additionally, the model specification repository can be used by more than one time-stamped data set.

The model specification repository contains the following information:

Model Specification File	SAS catalog entry or external file that *specifies a time series model to use* for forecasting.
Model Selection List File	SAS catalog entry or external file that specifies *how to select a model specification* to use for a particular time series.

The repository consists of SAS catalogs or external directories (or folders). More than one catalog can be combined using the SAS Libname Engine.

Creating a Model Specification Repository

You can create model specification files and populate the model specification repository using the HPFESMSPEC, HPFIDMSPEC, HPFARIMASPEC, HPFUCMSPEC, and HPFEXMSPEC procedures. After creating the model specification files, you can create model selection list files using the HPFSELECT procedure.

The following diagram illustrates the process of adding user-created model specification files and model selection list files:

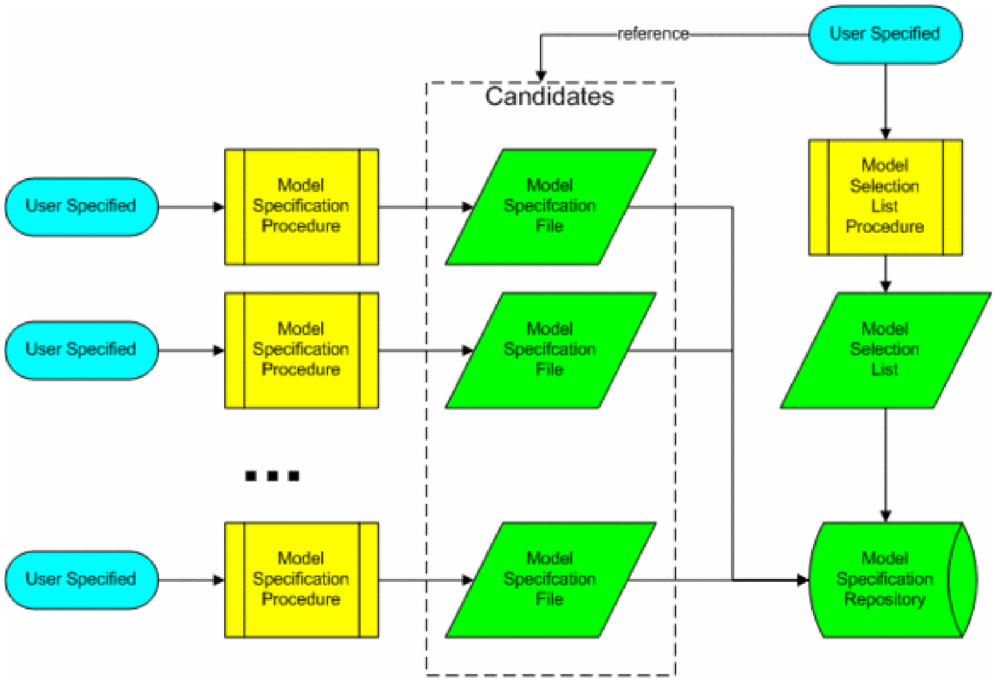


Figure 18.5. Creating a Model Specification Repository

You can also create the model specification files and model selection files using the HPFDIAGNOSE procedure. Given the historical time series data and the calendar events, the HPFDIAGNOSE procedure automatically creates model specification files and model selection files. The following diagram illustrates the series diagnostic process of automatically creating model specification files and model selection list files:

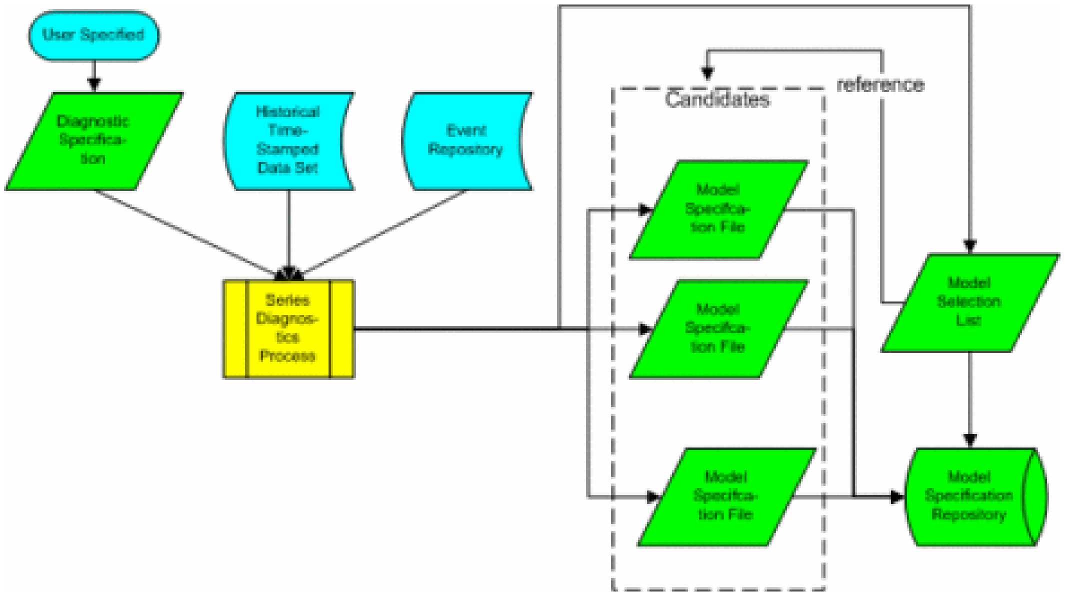


Figure 18.6. Series Diagnostic Process

Fitted Model Repository

A *fitted model repository* stores information about the selected model specification and its parameter estimates for each time series. Since each time series has different parameter estimates, the fitted model repository will often be large. There is one fitted model repository for each time-stamped data set.

The repository consists of a single SAS data set and associated SAS catalogs or external files that are referenced in the rows of the data set. The forecasting model repository is generated using the series diagnostics or default model selection lists.

For each time series, the fitted model repository specifies the following:

Model Selection List Name (reference)	SAS catalog entry name or external file name for the model selection list used to select this model. These lists are contained in a Model Specification Repository.
Model Specification Name (reference)	SAS catalog entry name or external file name that specifies the current model being used for forecasting. These specifications are contained in the Model Specification Repository.
Variable Mapping	Data set rows that map the time series data specification variables and events to model specification symbols.

Forecasting Model Parameter Estimates	Data set rows that contain the model parameter estimates associated with the current model.
Forecast Score Name (reference)	[*** need to clarify what this is ***] SAS catalog entry name or external file name that specifies the forecast scores associated with the current model. These scores are stored in the Forecast Score Repository.

Forecast Results Repository

A *forecast results repository* stores information about the forecasts, forecast evaluations, and forecast performance for each time series. The forecast results repository consists of several data sets. Since each time series has forecasts and statistics of fit associated with these forecasts, the forecast results repository will often be large. There is one forecast results repository for each time-stamped data set.

Score Repository

A *score repository* stores information about how to score each time series. Since each time series has a different score, the score repository will often be large because it summarizes information contained in the model specification repository, fitted model repository, as well as the final states (historical time series data). There is one score repository for each time-stamped data set.

Automatic Forecasting System Flow

Along with the time series data, the preceding information repositories are needed for large-scale automatic forecasting. Figure 18.7 shows the system flow for the automatic forecasting technique when there are many time series.

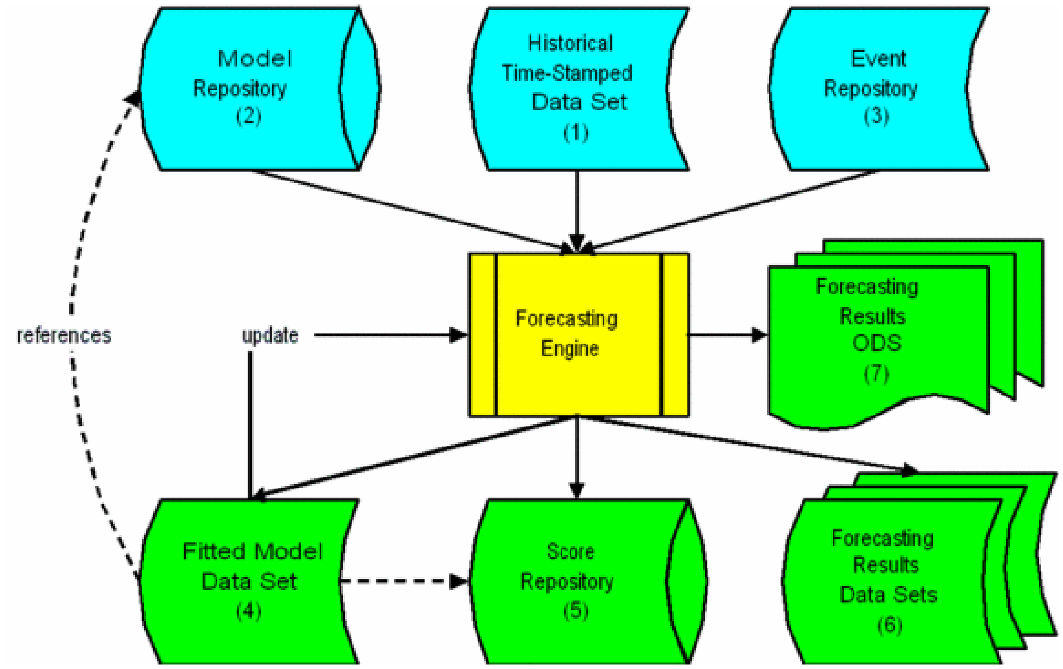

Model Repository (2)
Historical Time-Stamped Data Set (1)
Event Repository (3)
Forecasting Engine
Forecasting Results ODS (7)
references
update
Fitted Model Data Set (4)
Score Repository (5)
Forecasting Results Data Sets (6)

Figure 18.7. Automatic Forecasting System Flow

For each historical time series to forecast, the automatic forecasting system works as follows:

1. The time-stamp data are read from the time-stamped data set and accumulated, interpreted, and adjusted to form the time series to forecast.
2. The modeling information (model specifications and model selection list) associated with the time series is read from the model repository.
3. The calendar events associated with each model specification are read from the event repository.
4. Using the time series, modeling information, and calendar events, the forecasting engine creates or uses (updates) the fitted model.
5. From the fitted model, forecast score files are generated and stored in the score repository.
6. From the fitted model, forecast results data sets are created and stored.
7. From the fitted model, forecasting results ODS (printed tables and graphs) are created and rendered.

Forecast Scoring System Flow

For each time series, the automatic forecasting system generates a forecast score file that can be used in subsequent decision-making processes. Figure 18.8 shows the system flow for each file using the forecast scoring technique.

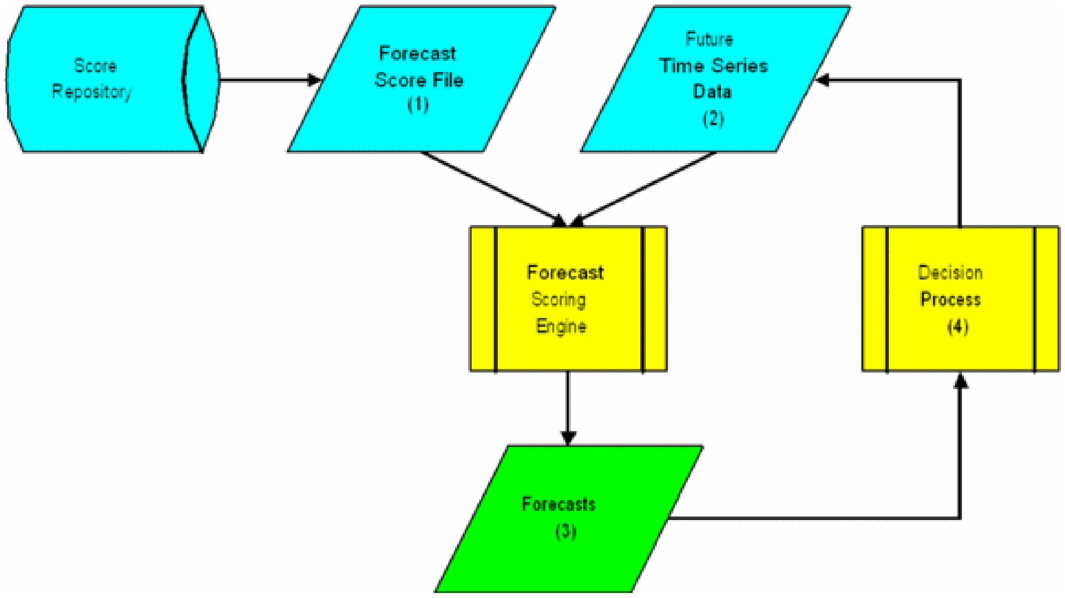


Figure 18.8. Forecast Scoring System Flow

For each time series to score, the forecast scoring process works as follows:

1. The forecast score file is read from the score repository.
2. The future values of the controllable causal factors are provided by the decision-making process.
3. Using the forecast score file and the future values, the forecast scoring process generates forecast results.
4. Steps 2 and 3 are repeated as needed by the decision-making process.

The automatic forecasting process that creates the forecast scoring file is significantly more computationally expensive than the forecast scoring process. The forecast score file only needs to be created once, whereas the iterative nature of decision-making processes may require many scores.

Automatic Forecasting Archives

Since automatic forecasting is used to forecast over time, the automatic forecasting process must be monitored for accuracy (quality control) or *forecast performance*. Therefore, the forecasts, generated over time, must be archived to measure forecast performance. Likewise, the forecast performance must be archived over time.

The automatic forecasting archives are shown in Figure 18.9.

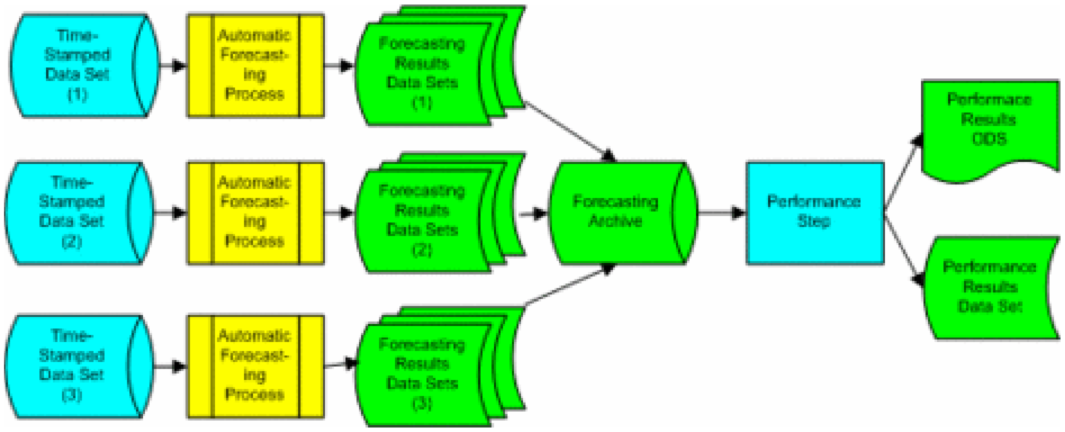

Time-Stamped Data Set (1)
Automatic Forecast-ing Process
Forecasting Results Data Sets (1)
Time-Stamped Data Set (2)
Automatic Forecast-ing Process
Forecasting Results Data Sets (2)
Time-Stamped Data Set (3)
Automatic Forecast-ing Process
Forecasting Results Data Sets (3)
Forecasting Archive
Performance Step
Performace Results ODS
Performance Results Data Set

Figure 18.9. Automatic Forecasting Archives

At each time the forecast is created (shown in the preceding diagram as 1, 2, 3) or *forecast origin*, forecasts are created from the time-stamped data observed up to the forecast origin. The forecasts from the forecast origin through the forecast horizon are recorded in the *forecasting archive*. The forecast archive contains the historical forecasts as well as their forecast origins. The forecast archive can be evaluated to measure the historical performance.

References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, Englewood Cliffs, N.J.:Prentice Hall, Inc.

Brockwell, P. J. and Davis, R. A. (1996), *Introduction to Time Series and Forecasting*, New York: Springer-Verlag.

Chatfield, C. (2000), *Time-Series Forecasting*, Boca Raton, FL: Chapman & Hall/CRC Press.

Fuller, W. A. (1995), *Introduction to Statistical Time Series*, New York: John Wiley & Sons, Inc.

Hamilton, J. D. (1994), *Time Series Analysis*, Princeton, NJ: Princeton University Press.

Harvey, A. C. (1994), *Time Series Models*, Cambridge, MA:MIT Press.

Makridakis, S. G., Wheelwright, S. C., and Hyndman, R. J. (1997), *Forecasting: Methods and Applications*, New York: John Wiley & Sons, Inc.

Chapter 19
Forecasting Process Details

Chapter Contents

Chapter 19
Forecasting Process Details

This chapter provides computational details on several aspects of the SAS High Performance Forecasting.

Forecasting Process Summary

This section summarizes the forecasting process. You can use a variety of forecasting models to forecast a series using SAS High Performance Forecasting. The final choice of model depends on the type of analysis needed and whether any predictor variables will be used in the forecasting or not. The available model types are, ARIMA models, UCM models, Smoothing models, and Intermittent models. The ARIMA and UCM models can utilize the predictor variables whereas the Smoothing and Intermittent models do not involve predictor variables.

Parameter Estimation

Computational details for the Smoothing and Intermittent models are provided in the sections "Smoothing Models" and "Intermittent Models." The details for ARIMA and UCM modeling are given in Chapter 11, “The ARIMA Procedure,” (*SAS/ETS User’s Guide*) and Chapter 30, “The UCM Procedure,” (*SAS/ETS User’s Guide*) , respectively. The results of the parameter estimation process are printed in the Parameter Estimates table or stored in the OUTEST= data set.

Model Evaluation

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 section "Statistics of Fit." The results generated by the evaluation process are printed in the Statistics of Fit table or stored in the OUTSTAT= data set.

Forecasting

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 printed in the Forecast table or stored in the OUTFOR= data set.

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.

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

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha + j\alpha\gamma & \text{for } j \bmod p \neq 0 \\ \alpha + j\alpha\gamma + \delta(1-\alpha), & \text{for } j \bmod p = 0 \end{cases}$$

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

HPF uses the same statistical model technology to identify, fit and forecast ARIMA models as does SAS/ETS Software. Refer to Chapter 11, "The ARIMA Procedure," (*SAS/ETS User's Guide*) for details on the methods HPF uses for ARIMA models. All the SAS/ETS ARIMA output such as the parameter estimates, forecasts, diagnostic measures, etc, is available in HPF. Moreover, you can obtain additional output in HPF. This includes a wider variety of fit statistics and a model based decomposition of the response series forecasts in to subcomponents such as transfer functions effects and the estimated stationary noise component. These subcomponents can be useful in the interpretation of the model being used.

ARIMA Model Based Series Decomposition

Consider a general ARIMA model that may be fit to a response series Y_t:

$$D(B)Y_t = \mu + \sum_i \frac{\omega_i(B)}{\delta_i(B)} B^{k_i} X_{i,t} + \frac{\theta(B)}{\phi(B)} a_t$$

where

t	indexes time
B	is the backshift operator; that is, $BX_t = X_{t-1}$
$D(B)$	is the difference operator operating on the response series Y_t
μ	is the constant term
$\phi(B)$	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
$X_{i,t}$	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 expresses the response series, possibly differenced, as a sum of a constant term, various transfer function effects, and the effect of a stationary ARMA disturbance term. Of course, in a given situation many of these effects might be absent. Denoting the individual transfer function effects by γ_i and the ARMA disturbance term by n, we can decompose the response series in a few different ways. Here we will consider two alternate decompositions. In the first case Y_t is decomposed as

$$Y_t = L_t + \mu + \sum_i \gamma_{it} + n_t$$

where the L_t term includes the contribution from the lagged response values corresponding to the differencing operator $D(B)$, e.g., if $D(B) = 1 - B$, corresponding to the differencing of order 1, then $L_t = Y_{t-1}$. An alternate decomposition of Y_t can be as follows:

$$Y_t = \frac{\mu}{D(B)} + \sum_i \frac{\gamma_{it}}{D(B)} + \frac{n_t}{D(B)}$$

Note that if the differencing operator $D(B)$ is identity then the L_t term is zero and these two alternate decompositions are identical, otherwise the terms in the second decomposition are the "integrated", with respect to $D(B)$, versions of the corresponding terms in the first decomposition. In practice many terms in these decompositions are not observed but are estimated from the data, which results in similar decompositions of $\hat{Y}_t$, the *forecasts* of Y_t. In the HPFENGINE procedure you can obtain these decompositions of $\hat{Y}_t$ by choosing various options in the PROC HPFENGINE statement: you can output them as a data set using the OUTCOMPONENT= data set option, print them using the PRINT=COMPONENT option, or plot them using the PLOT=COMPONENTS option. The type of decomposition is controlled by the COMPONENTS= option in the HPFENGINE statement. If COMPONENTS=INTEGRATE option is specified then the calculated decomposition is of the second type, the default decomposition is of the first type. A few points to note about these decompositions:

- If the response series being modeled is actually $\log$ transformed then the resulting decompositions are *multiplicative* rather than *additive*. In this case, similar to the series forecasts, the decomposition terms are also inverse transformed. If the response series is transformed using a transformation other than $\log$, such as Box-Cox or Square Root, then these decompositions are difficult to interpret and they do not have such additive or multiplicative properties.
- In the first type of decomposition the components in the decomposition will always add up, or multiply in the $\log$ transformation case, to the series forecasts. In the integrated version of the decomposition this additive property may not always hold because there are no natural choices of starting values that can be used during the integration of these components that guarantee the additivity of the resulting decomposition.

UCM Models

HPF uses the same statistical model technology to identify, fit and forecast UCM models as does SAS/ETS Software. Refer to Chapter 30, "The UCM Procedure," (*SAS/ETS User's Guide*) for details on the methods HPF uses for UCM models.

Intermittent Models

This section details the computations performed for intermittent forecasting models.

Intermittent Time Series

Intermittent time series have a large number of values that are zero. These types of series commonly occur in Internet, inventory, sales, and other data where the demand for a particular item is intermittent. Typically, when the value of the series associated with a particular time period is nonzero, *demand* occurs; and, when the value is zero (or missing), *no demand* occurs. Since it is entirely possible that the number of time periods for which *no demand* occurs is large, many of the series values will be zero. Typical time series models (for example, smoothing models) are inadequate in the case of intermittent time series because many of the series values are zero. Since these models are based on weighted-summations of past values, they bias forecasts away from zero. Unlike the smoothing models that provide forecasts for future time periods, intermittent forecasting models provide recommended *stocking levels* or *estimated demand per period* that are used to satisfy future demand.

Intermittent Series Decomposition and Analysis

An intermittent time series (demand series) can be decomposed into two components: a demand interval series and a demand size series. Both of these component series are indexed based on when a *demand* occurred (demand index) rather than each time period (time index). The demand interval series is constructed based on the number of time periods between *demands*. The demand size series is constructed based on the size (or value) of the *demands* excluding zero (or base) demand values. Using these two component series, the average demand series is computed by dividing the size component values by the interval component values.

When a *demand* occurs typically depends on a *base* value. Typically, the base value is zero (default), but it can be any constant value and can be automatically determined based on the characteristics of the demand series.

Given a time series y_t, for $t = 1$ to T, where t is the time index, suppose that there are N nonzero demands occurring at times $t = t_i$, where $t_{i-1} < t_i$, for $i = 1$ to N. The time series is dissected into the demand interval series and the demand size series as follows:

(Demand Interval Series)	$q_i = t_i - t_{i-1}$	for $i = 2$ to N
(Demand Size Series)	$d_i = y_{t_i} - base$	for $i = 1$ to N
(Average Demand Series)	$a_i = d_i / q_i$	for $i = 2$ to N

For the beginning of the demand series, q_1 is assigned to t_1, which assumes that a demand just occurred prior to the first recorded observation. For the end of the demand series, q_{N+1} is assigned to $(T + 1 - t_N)$, which assumes that demand will occur just after the last recorded observation. The next future demand size, d_{N+1}, is always set to missing.

After decomposition, descriptive (summary) statistics can be computed to gain a greater understanding of the demand series including those statistics based on the season index.

For statistical analysis and model fitting, q_i and a_i for $i = 2$ to N and d_i for $i = 1$ to N are used. For forecasting, q_i for $i = 1$ to $N + 1$, a_i for $i = 1$ to N, d_i for $i = 1$ to N are used.

Croston's Method

Croston's Method models and forecasts each component independently, then combines the two forecasts. The following provides a description of how Croston's Method is used in SAS High Performance Forecasting. More detailed information on this method can be found in Croston (1972) and Willemain, Smart, and Shocker (1994). The following description of Croston's Method is based on the perspective of a person familiar with typical time series modeling techniques such as smoothing models.

By treating each component of the demand series as a time series based on the demand index, optimal smoothing parameters can be estimated and predictions for each component can be computed using nonseasonal exponential smoothing methods (simple, double, linear, damped-trend) as well as their transformed versions (log, square-root, logistic, Box-Cox).

For example, the following simple smoothing equations are used to generate predictions for demand size and interval components:

$$\begin{array}{ll}\text{(Smoothed demand interval series)} & L_i^q = L_{i-1}^q + \alpha_q(q_{i-1} - L_{i-1}^q) \\ \text{(Smoothed demand size series)} & L_i^d = L_{i-1}^d + \alpha_d(d_{i-1} - L_{i-1}^d)\end{array}$$

The demand interval parameter, α_q, and demand size parameter, α_d, and the starting, intermediate, and final smoothing level states, L_i^q and L_i^d, are estimated from the data using simple exponential smoothing parameter estimation. For the starting state at $i = 1$, $L_1^q = \max(q_1, L_0^q)$ where L_0^q is the final backcast level state. For $i > 1$, the one-step-ahead prediction for demand interval q_i is $\hat{q}_i = L_{i-1}^q$. For $i > 0$, the one-step-ahead prediction for demand size d_i is $\hat{d}_i = L_{i-1}^d$.

Other (transformed) nonseasonal exponential smoothing methods can be used in a similar fashion. For linear smoothing, $L_1^q = \max(q_1 - T_0^q, L_0^q)$ where T_0^q is the final backcast trend state. For damp-trend smoothing, $L_1^q = \max(q_1 - \phi_q T_0^q, L_0^q)$ where ϕ_q is the damping parameter and T_0^q is the final backcast trend state. For double smoothing, $L_1^q = \max(q_1 - T_0^q/\alpha_q, L_0^q)$ where α_q is the weight parameter and T_0^q is the final backcast trend state.

Using these predictions based on the demand index, predictions of the average demand per period can be estimated. Predicted demand per period is also known as "stocking level," assuming that disturbances affecting q_i are independent of d_i. (This assumption is quite significant.)

(Estimated demand per period)	$y_i^* = \hat{d}_i / \hat{q}_i$
	$\mathrm{E}[y_i^*] = E[d_i]/E[q_i] = E[z_{i-1}]/E[p_{i-1}] = \mu_d/\mu_q$
(Variance)	$Var(y_i^*) = (\hat{d}_i/\hat{q}_i)^2(Var(d_i)/\hat{d}_i^2 + Var(q_i)/\hat{q}_i^2)$

where μ_d, $\bar{d}$, and s_d are the mean, sample average, and standard deviation of the non-zero demands, and μ_q, $\bar{q}$, and s_q are the mean, sample average, and standard deviation of the number of time periods between demands.

For the beginning of the series, the denominator of y_1^* is assigned q_i or the starting smoothing state p_0, whichever is greater. For the end of the series, the denominator of y_{N+1}^* is assigned $q_{N+1} = (T+1-t_N)$ or the final smoothing state p_N, whichever is greater.

Once the average demand per period has been estimated, a stocking level can be recommended:

(Recommended stocking level)	$\hat{y}_t = y_i^*$	when $t_i =< t < t_{i+1}$
(Variance)	$Var(\hat{y}_t) = Var(y_i^*)$	when $t_i =< t < t_{i+1}$

Since the predicted demand per period will be different than typical time series forecasts, the usual way of computing statistics of fit should also be different. The statistics of fit are based on the difference between the recommended stocking levels between demands and the demands:

(Accumulated recommended stocks)	$s_t = \sum_{i=0}^{t}(\hat{y}_t - y_i^*)$	
(Estimate – Demand)	$e_{t_i} = \hat{d}_i q_i - d_i$	when time t_{i+1} has demand

Croston's Method produces the same forecasts as simple exponential smoothing when demand occurs in every time period, $q_i = 1$ for all i, but different (lower) prediction error variances. Croston's Method is recommended for intermittent time series only.

Average Demand Method

Similar to Croston's Method, the Average Demand Method is used to forecast intermittent time series; however, the Average Demand Method forecasts the average demand series directly, whereas Croston's Method forecasts average demand series indirectly using the inverse decomposition of the demand interval and size series forecasts.

By treating the average demand series as a time series based on the demand index, optimal smoothing parameters can be estimated and predictions for average demand can be computed using nonseasonal exponential smoothing methods (simple, double, linear, damped-trend) as well as their transformed versions (log, square-root, logistic, Box-Cox).

For example, the following simple smoothing equations are used to generate predictions for the average demand series:

(Smoothed Average Demand Series) $L^a_i = L^a_{i-1} + \alpha_a(a_{i-1} - L^a_{i-1})$

The average demand level smoothing parameter, α_a, and the starting, intermediate, and final smoothing level states, L^a_i, are estimated from the data using simple exponential smoothing parameter estimation. For the starting state at $i = 1$, $L^a_1 = \max(a_1, L^a_0)$ where L^a_0 is the final backcast level state. For $i > 1$, the one-step-ahead prediction for a_i is $\hat{a}_i = L^a_{i-1}$.

Other nonseasonal exponential smoothing methods can be used in a similar fashion. For linear smoothing, $L^a_1 = \max(a_1 - T^a_0, L^a_0)$ where T^a_0 is the final backcast trend state. For damp-trend smoothing, $L^a_1 = \max(a_1 - \phi_a T^a_0, L^a_0)$ where ϕ_a is the damping parameter and T^a_0 is the final backcast trend state. For double smoothing, $L^a_1 = \max(a_1 - T^a_0/\alpha_a, L^a_0)$ where α_a is the weight parameter and T^a_0 is the final backcast trend state.

Using these predictions based on the demand index, predictions of the average demand per period are provided directly, unlike Croston's Method where the average demand is predicted using a ratio of predictions of the demand interval and size components.

(Estimated demand per period)	$y^*_i = \hat{a}_i + base$
	$\mathrm{E}[y^*_i] = E[a_i]$
(Variance)	see the exponential smoothing models

For the beginning of the series, $\hat{a}_1$ is derived from the starting level smoothing state and starting trend smoothing state (if applicable).

Once the average demand per period has been estimated, a stocking level can be recommended similar to Croston's Method.

The Average Demand Method produces the same forecasts as exponential smoothing when demand occurs in every time period, $q_i = 1$ for all i, but different (lower) prediction error variances. The Average Demand Method is recommended for intermittent time series only.

Time-Indexed versus Demand-Indexed Holdout Samples

Holdout samples are typically specified based on the time index, but for intermittent demand model selection, demand indexed-based holdouts are used for model selection.

For example, "holdout the last six months data." For a demand series, the demand indexed holdout refers to the "demands that have occurred in the last six months." If there are four demands in the last six months, the demand indexed holdout is four for a time indexed holdout of six. If there are no demands in the time indexed holdout, the demand indexed holdout is zero and in-sample analysis is used.

Automatic Intermittent Demand Model Selection

The exponential smoothing method to be used to forecast the intermittent demand series can be specified, or it can be selected automatically using a model selection criterion and either in-sample or holdout sample analysis. The exponential smoothing method for each demand series component (interval, size, and average) can be automatically selected as well as the choice between Croston's Method and the Average Demand Method.

For Croston's Method, the exponential smoothing methods used to forecast the demand interval and size components are automatically selected independently. For the Average Demand Method, the exponential smoothing methods used to forecast the average demand component are automatically selected, again independently. Based on the model selection criterion, the selection is based on how well the method fits (in sample) or predicts (holdout sample) the demand series component by treating the demand index as a time index. The following equations describe the component prediction errors associated with each of the demand series components that are used in component model selection:

(Demand Interval Series)	$e_i^q = q_i - \hat{q}_i$	for $i = 2$ to N
(Demand Size Series)	$e_i^d = d_i - \hat{d}_i$	for $i = 1$ to N
(Average Demand Series)	$e_i^a = a_i - \hat{a}_i$	for $i = 2$ to N

Once the exponential smoothing methods are selected for each demand series component, the predictions for either Croston's Method, $(\hat{d}_i/\hat{q}_i)$, the Average Demand Method, $\hat{a}_i$, or both are computed based on the selected method for each component.

When choosing between Croston's Method and the Average Demand Method, the model is selected by considering how well the model predicts average demand with respect to the time. The following equations describe the average prediction errors associated with the predicted average demand that are used in model selection:

(Croston's Method)	$e_i^c = (d_i/q_i) - (\hat{d}_i/\hat{q}_i)$	for $i = 2$ to N
(Average Demand Method)	$e_i^a = a_i - \hat{a}_i$	for $i = 2$ to N

External Models

External forecasts are forecasts provided by an *external source*. External forecasts may originate from a statistical model, from another software package, may have been provided by an outside organization (e.g. marketing organization

, government agency), or may be given based solely judgment.

External Forecasts

Given a time series, $\{y_t\}_{t=1}^{T}$, where $t = 1, \ldots, T$ is the time index and T is the length of the time series, the external model data source must provide predictions for the future time periods, $\{\hat{y}_t\}_{t=T+1}^{T+H}$, where H

represents the forecast horizon. The external data source may or may not provide in-sample predictions for past time periods, $\{\hat{y}_t\}_{t=1}^{T}$. Additionally, the external source may or may not provide the prediction standard errors, $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$, lower confidence limits, $\{Lower(\hat{y}_t)\}_{t=1}^{T+H}$, and the upper confidence limits, $\{Upper(\hat{y}_t\}_{t=1}^{T+H}$, for the past or future time periods.

External Forecast Prediction Errors

If the external forecast predictions are provided for past time periods, the external forecast prediction errors, $\{\hat{e}_t\}_{t=1}^{T}$, can be computed, $\hat{e}_t = y_t - \hat{y}_t$. If any of these predictions are not provided by the external source

, the prediction errors are set to missing.

For judgmental forecast with no historical judgments, all prediction errors will be missing. For this situation, a judgment about the prediction standards errors should also be provided.

External Forecast Prediction Bias

It is often desirable that forecasts be unbiased. If available, the external forecast prediction errors are used to compute prediction bias from the external source.

When the external forecast predictions are provided for past time periods, HPF uses the in-sample prediction errors $\{\hat{e}_t\}_{t=1}^{T}$ to estimate the bias of the forecasts provided by the external source.

$$Bias = \sum_{t=1}^{T} (y_t - \hat{y}_t)/T = \sum_{t=1}^{T} \hat{e}_t/T$$

The prediction mean square error can be used to help judge the significance of this bias.

$$Variance = \sum_{t=1}^{T} (\hat{e}_t - Bias)^2/(T-1)$$

Missing values are ignored in the above computations and the denominator term is reduced for each missing value.

External Forecast Prediction Standard Errors

The external model data may include estimates of the prediction standard errors and confidence limits. If these estimates are not supplied with the data for the external model (which is usually the case for judgmental forecasts), HPF can approximate the

prediction standard errors and the confidence limits using the in-sample prediction errors.

When the external forecasts do not contain the prediction standard errors, they are approximated using the external forecast prediction errors. In order to approximate the external forecast prediction standard errors, the external model residuals, variance, and autocorrelations must be approximated. (If the prediction standard errors are provided by the external source, approximation is not requi

red.)

In order to approximate the external model residuals, $\{\hat{\epsilon}_t\}_{t=1}^{T}$, the transformation used to generate the external forecast must be provided as part of the external model specification. Let $z_t = f(y_t)$ be the specified transformation; and if there is no specified functional transformation, $z_t = y_t$. The transformation can be used to approximate the external model residuals, $\hat{\epsilon}_t = f(y_t) - f(\hat{y}_t) = z_t - \hat{z}_t$.

Once approximated, the external model residuals can be used to approximate the external model variance, $\hat{\sigma}_\epsilon^2 = \frac{1}{T}\sum_{t=1}^{T} \hat{\epsilon}_t^2$, and the external model residual autocorrelation.

The external model residual autocorrelations can be approximated given assumptions about the autocorrelation structure, $\rho(j)$ where $j \geq 0$ represents the time lags.

The following autocorrelation structures can be specified using the METHOD= option. These options are listed in order of increasing assumed autocorrelation. (In the following formula, $0 < v < 1$ represents the value of the NLAGPCT= option.)

No Autocorrelation (METHOD=NONE)

$$\rho(j) = 1 \quad \text{for} \quad j = 0$$

$$\rho(j) = 0 \quad \text{for} \quad j > 0$$

This option generates constant prediction error variances.

White Noise Autocorrelation (METHOD=WN)

This option generates white noise prediction error variances.

Error Autocorrelation (Method=ERRORACF)

$$\rho(j) = 1 \quad \text{for} \quad j = 0$$

$$\hat{\rho}(j) = \frac{1}{T}\sum_{t=j+1}^{T} \hat{\epsilon}_t \hat{\epsilon}_{t-j} / \hat{\sigma}_\epsilon^2 \quad \text{for} \quad 0 < j < vT$$

$$\rho(j) = 0 \quad \text{for} \quad j > vT$$

Series Autocorrelation (METHOD=ACF)

$$\rho(j) = 1 \qquad \text{for} \qquad j = 0$$

$$\hat{\rho}(j) = \tfrac{1}{T}\sum_{t=j+1}^{T} z_t z_{t-j}/\hat{\sigma}_z^2 \qquad \text{for} \qquad 0 < j < vT$$

$$\rho(j) = 0 \qquad \text{for} \qquad j > vT$$

where $\hat{\sigma}_z^2 = \frac{1}{T}\sum_{t=1}^{T}(z_t - \bar{z})^2$ and $0 < v < 1$.

This option typically generates linear prediction error variances larger than ERRORACF.

Perfect Autocorrelation (METHOD=PERFECT)

$$\rho(j) = 1 \qquad \text{for} \qquad j \geq 0$$

This option generates linear prediction error variances.

The external model residual variance and the autocorrelations can approximate the external (transformed) prediction standard errors, $\{Std(\hat{z}_t)\}_{t=1}^{T+H}$, where $Std(\hat{z}_t) = \sigma_\epsilon$ for $t = 1, \ldots, T$ are the one-step-ahead prediction standards errors and $Std(\hat{z}_{T+h}) = \sqrt{\sigma_\epsilon^2 \sum_{j=0}^{h-1} \hat{\rho}(j)^2}$ for $h = 1, \ldots, H$ are the multi-step-ahead prediction standard errors.

If there was no transformation used to generate the external forecasts, $Std(\hat{y}_t) = Std(\hat{z}_t)$. Otherwise, the external transformed predictions, $\{\hat{z}\}_{t=1}^{T}$, and the approximate external transformed prediction standard errors, $\{Std(\hat{z}_t)\}_{t=1}^{T+H}$, are used to approximate the external prediction standard errors, $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$, by computing the conditional expectation of the inverse transformation (mean) or computing the inverse transformation (median).

To summarize, the external forecast prediction standard errors, $\{Std(\hat{y}_t)\}_{t=1}^{T+H}$, can be approximated from the external forecast prediction errors, $\{\hat{e}_t\}_{t=1}^{T}$, given the following assumptions about the external forecasts provided by the external source:

Functional Transformation	(TRANSFORM= option)	$z_t = f(y_t)$
Autocorrelation Structure	(METHOD= option)	$\rho(j)$
Autocorrelation Cutoff	(NLAGPCT= option)	v

External Forecast Confidence Limits

Since the external forecasts may not contain confidence limits, they must be approximated using the forecast prediction standard errors (which may be provided by the external model data source or approximated from the in-sample prediction errors).

$$Lower(\hat{y}_t) = \hat{y}_t - Std(\hat{y}_t)Z_{\frac{\alpha}{2}}$$

$$Upper(\hat{y}_t) = \hat{y}_t + Std(\hat{y}_t)Z_{\frac{\alpha}{2}}$$

where α is the confidence limit width, $(1-\alpha)$ is the confidence level, and $Z_{\frac{\alpha}{2}}$ is the $\frac{\alpha}{2}$ quantile of the standard normal distribution.

To summarize, the external forecast confidence limits can be approximated given only the actual time series, $\{y_t\}_{t=1}^{T}$, and the external forecast predictions, $\{\hat{y}_t\}_{t=T+1}^{T+H}$. More information provided about the external forecast prediction standard errors, $\{Std(y_t)\}_{t=1}^{T+H}$, such as the functional transform, $f()$, and the autocorrelation structure, $\rho(j)$ and v, improves the accuracy of these approximations.

Series Transformations

For forecasting models, transforming the time series may aid in improving forecasting accuracy.

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 $$c = (1-e^{-6})10^{-\mathrm{ceil}(\log_{10}(\max(\mathrm{y_t})))}$$ 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 $$w_t = \begin{cases} \frac{y_t^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(\mathrm{y_t}), & \lambda = 0 \end{cases}$$

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 $c = (1 - 10^{-6})10^{-\mathrm{ceil(log_{10}(max(y_t)))}}$.

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.

$$\hat{y}_t = \begin{cases} E\left[(\lambda w_t + 1)^{1/\lambda}\right], & \lambda \neq 0 \\ E\left[e^{w_t}\right] = \exp(\mathrm{\hat{w}_t} + \frac{1}{2}\sigma_\mathrm{t}^2), & \lambda = 0 \end{cases}$$

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.

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 or whether or not the time series is intermittent. These recommendations are used by the automatic model selection process to restrict the model search to a subset of the model selection list.

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.

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 SAS High Performance Forecasting. The testing strategy is as follows:

1. **Intermittent test.** Compute the average time interval between demands. If the time average time interval is greater than a preset limit, an intermittent forecasting model is used.
2. **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. nonseasonal model, a seasonal forecasting model is used.

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 printed or stored in a data set.

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. $APE = |100 * (y_t - \hat{y}_t)/y_t|$ is the absolute percent error. $ASPE = |100 * (y_t - \hat{y}_t)/0.5(y_t + \hat{y}_t)|$ is the absolute symmetric percent error. $APPE = |100 * (y_t - \hat{y}_t)/\hat{y}_t|$ is the absolute predictive percent error. $RAE = |(y_t - \hat{y}_t)/(y_t - y_{t-1})|$ is the relative absolute error. The errors are ignored in the statistical computations when the denominator is zero.

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 Error.
The mean absolute prediction error, $\frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|$.

Minimum Absolute Percent Error.
The minimum of the absolute percent errors (MINAPE).

Maximum Absolute Percent Error.
The maximum of the absolute percent errors (MAXAPE).

Mean Absolute Percent Error.
The mean of the absolute percent errors (MAPE).

Median Absolute Percent Error.
The median of the absolute percent errors (MdAPE).

Geometric Mean Absolute Percent Error.
The geometric mean of the absolute percent errors (GMAPE).

Minimum Absolute Symmetric Percent Error.
The minimum of the absolute symmetric percent errors (MINASPE).

Maximum Absolute Symmetric Percent Error.
The maximum of the absolute symmetric percent errors (MAXASPE).

Mean Absolute Symmetric Percent Error.
The mean of the absolute symmetric percent errors (MASPE).

Median Absolute Symmetric Percent Error.
The median of absolute symmetric percent errors (MdASPE).

Geometric Mean Symmetric Percent Error.
The geometric mean of the absolute symmetric percent errors (GMASPE).

Minimum Absolute Predictive Percent Error.
The minimum of the absolute predictive percent errors (MINAPPE).

Maximum Absolute Predictive Percent Error.
The maximum of the absolute predictive percent errors (MAXAPPE).

Mean Absolute Predictive Percent Error.
The mean of the absolute predictive percent errors (MAPPE).

Median Absolute Predictive Percent Error.
The median absolute predictive percent prediction error (MdAPPE).

Geometric Mean Absolute Predictive Percent Error.
The geometric mean absolute predictive percent prediction error (GMAPPE).

Minimum Relative Absolute Error.
The minimum of the relative absolute errors (MINRAE).

Maximum Relative Absolute Error.
The maximum of the relative absolute errors (MAXRAE).

Mean Relative Absolute Error.
The mean of the relative absolute errors (MRAE).

Median Relative Absolute Error.
The median of the relative absolute errors (MdRAE).

Geometric Relative Absolute Error.
The geometric mean of the relative absolute errors (GMRAE).

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), $n \ln(SSE/n) + 2k$.

Schwarz Bayesian Information Criterion.
Schwarz Bayesian information criterion (SBC or BIC),
$n \ln(SSE/n) + k \ln(n)$.

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

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.

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.

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.

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 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.

Croston, J.D. (1972), "Forecasting and Stock Control for Intermittent Demands," *Operations Research Quarterly*, 23, No. 3.

Davidson, J. (1981), "Problems with the Estimation of Moving Average Models," *Journal of Econometrics*, 16, 295.

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.

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.

Pankratz, A. and Dudley, U. (1987), "Forecast of Power-Transformed Series," *Journal of Forecasting*, Vol 6, No. 4, 239-248.

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.

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.

Willemain, T.R, Smart, C.N, Shocker, J.H (1994) “Forecasting Intermittent Demand In Manufacturing: Comparative Evaluation of Croston’s Method,” *International Journal of Forecasting*, 10, 529-538.

Chapter 20
Using Scores (Experimental)

Chapter Contents

Chapter 20
Using Scores (Experimental)

Overview

The HPFENGINE procedure can generate forecast score files. The subroutines described in this chapter are used in conjunction with these score files to produce forecasts.

HPFSCSIG Function

The HPFSCSIG function creates a text string representing a template for the appropriate usage of HPFSCSUB given a forecast scoring file.

Syntax

HPFSCSIG(*Scoring-XML-fileref, horizon, ReturnType* **)**

Scoring-XML-Fileref	is a SAS file reference that contains the forecast scoring information.
Horizon	is the forecast horizon or lead. This must be a positive integer value.
ReturnType	is one of the following strings: PREDICT, STDERR, LOWER, or UPPER. This determines whether the score function will compute and return the forecast value, standard error, lower confidence limit, or upper confidence limit, respectively.

Details

The syntax of the forecast scoring function is variable and depends on the horizon and certain details found in the score file. HPFSCSIG aids the user in constructing subroutine calls with the correct syntax.

Examples

Consider the following case of an ARIMAX model with three inputs. Two of them are predefined trend curves and the third is specified as controllable in the call to PROC HPFENGINE. A score file is produced and HPFSCSIG called to provide a template for the call of HPFSCSUB.

```
proc hpfarimaspec modelrepository=work.repo specname=ar;
   dependent symbol=Y q=1 dif=12;
   input predefined=LINEAR;
   input symbol=controlinput;
   input predefined=INVERSE;
run;

proc hpfselect modelrepository=work.repo selectname=select;
   spec ar;
run;

proc catalog catalog=work.scores kill; run;

proc hpfengine data=air print=(select estimates)
      modelrepository=work.repo globalselection=select
      outest=engineoutest scorerepository=work.scores;
   id date interval=month;
   forecast air;
   controllable controlinput / extend=avg;
   score;
run;

filename score catalog "work.scores.scor0.xml";

data _null_;
   sig = hpfscsig('score',3, 'predict');
   put sig=;
run;
```

This function produces the following output on the SAS log:

```
sig=HPFSCSUB('XML',3,'CONTROLINPUT',?,?,?,'PREDICT',!,!,!)
```

In place of XML, the user will provide the pre-assigned SAS fileref. The user will also replace the question marks (?) with the desired inputs and the output will be written to variables placed where there are exclamation marks (!).

HPFSCSUB Function

The HPFSCSUB subroutine returns the forecasts, standard errors, or confidence limits given a forecast scoring file and future values of all controllable inputs.

Syntax

HPFSCSUB(*Scoring-XML-fileref, horizon, X1, input-1-1, ... , input-1-horizon, Xj, input-j-1, ... , input-j-horizon, OutputType, output-1, ... , output-horizon* **)**

Scoring-XML-Fileref	is a SAS file reference that contains the forecast scoring information.
Horizon	is the forecast horizon or lead. This must be a positive integer value.

X_j	indicates that the next horizon values are the future inputs for controllable variable j.
Input-j-k	is a controllable input value for the *j*-th variable at horizon *k*.
OutputType	one of the following: PREDICT, STDERR, LOWER, UPPER. Indicates what will be returned in the output variables.
Output-k	the subroutine output at horizon *k*.

Details

The HPFSCSUB subroutine returns the forecasts, standard errors, or confidence limits given a forecast scoring file and future values of all controllable inputs. Because the syntax is variable and depends on the input score file, HPFSCSIG is normally used to determine the layout of the subroutine call.

The score might have been computed using and ARIMAX, UCM or another model. HPFSCSUB automatically determines detects this and computes the requested return values appropriately.

Examples

This example uses a score file to compute a forecast. The score file is stored in the scor0 entry within the catalog work.score. Note that even though the model includes three inputs, only one is designated controllable in the call to PROC HPFENGINE. Therefore, only future values of the variable controlinput are required to generate forecasts using the score file.

In the call to PROC HPFENGINE, the controllable input was extended with the mean of the controlinput series. Therefore the mean is used as input to the forecast score function so that a valid comparison can be made between the forecast results from PROC HPFENGINE and HPFSCSUB.

```
proc hpfarimaspec modelrepository=work.repo specname=ar;
   dependent symbol=Y q=1 dif=12;
   input predefined=LINEAR;
   input symbol=controlinput;
   input predefined=INVERSE;
run;

proc hpfselect modelrepository=work.repo selectname=select;
   spec ar;
run;

proc catalog catalog=work.scores kill; run;

* generate score;
proc hpfengine data=air modelrepository=work.repo out=engineout
      globalselection=select scorerepository=work.scores;
   id date interval=month;
   forecast air;
   controllable controlinput / extend=avg;
   score;
```

```
run;

filename score catalog "work.scores.scor0.xml";

proc means data=air mean noprint;
   var controlinput;
   output out=controlmean mean=mean;
run;

data _null_;
   set controlmean;
   call symput("mean", mean);
run;

data forecasts;
   drop p1 p2 p3;
   format date monyy.;
   date = '01jan1961'd;
   call HPFSCSUB('score',3,'CONTROLINPUT',&mean,&mean,&mean,
                 'PREDICT',p1,p2,p3);
   forecast = p1; date = intnx('month', date, 0); output;
   forecast = p2; date = intnx('month', date, 1); output;
   forecast = p3; date = intnx('month', date, 1); output;
run;

data compare;
   merge engineout forecasts;
   by date;
run;

proc print data=compare(where=(forecast ne .)) noobs;
run;
```

The output is

```
     DATE       AIR      forecast

  JAN1961    416.408     416.408
  FEB1961    391.715     391.715
  MAR1961    419.312     419.312
```

Chapter 21
User-Defined Models

Chapter Contents

Chapter 21
User-Defined Models

Introduction

User-defined forecasting models can be used in SAS High-Performance Forecasting. The forecasts produced by these models are considered external forecasts because they are forecasts originating from an external source.

A user-defined forecasting model can be written in the SAS language or the C language by using the FCMP procedure or the PROTO procedure, respectively. The HPFENGINE procedure cannot use C language routines directly. The procedure can only use SAS language routines that may or may not call C language routines. Creating user-defined routines are more completely described in the FCMP procedure and the PROTO procedure documentation. For more information about the FCMP and PROTO procedures, see the Base SAS Community at http://support.sas.com/documentation/onlinedoc/base.

The SAS language provides integrated memory management and exception handling such as operations on missing values. The C language provides flexibility and allows the integration of existing C language libraries. However, proper memory management and exception handling are solely the responsibility of the user. Additionally, the support for standard C libraries is restricted. If given a choice, it is highly recommended that user-defined functions and subroutines and functions be written in the SAS language using the FCMP procedure.

In order to use a SAS language function or routine, an external model specification must be specified as well. In order to use a C or C++ language external function, it must be called by a SAS language function or subroutine and an external model specification must be specified as well. External model specifications are specified by the HPFEXMSPEC procedure. The following diagram describes an example of the ways user-defined forecasting models can be defined, specified and used to forecast.

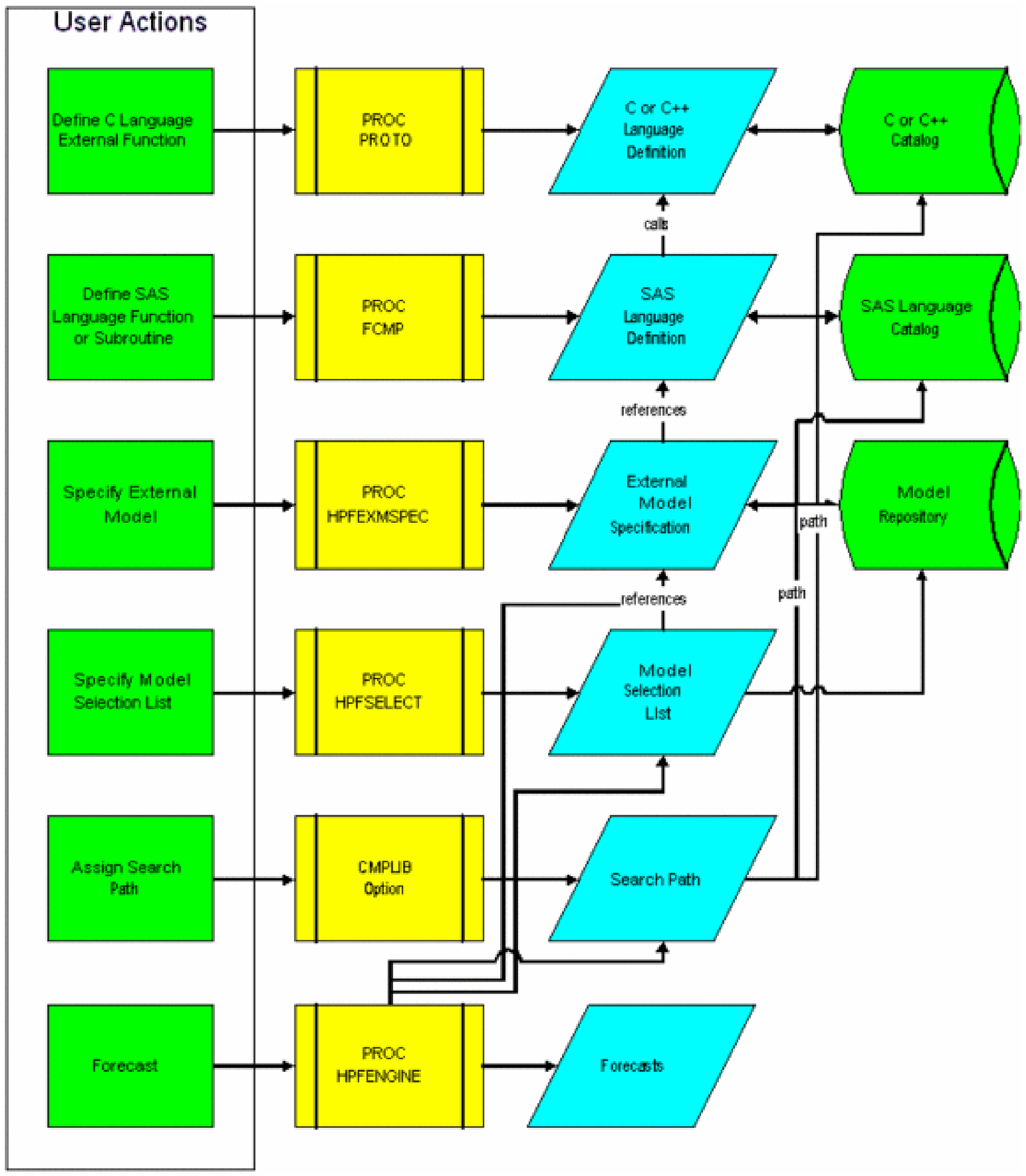


Figure 21.1. User-defined Forecasting Models

The SAS language function or routine can call other SAS language functions or routines provided that the search path for these functions and routines are provided.

Defining and Using a SAS Language Function or Subroutine

The FCMP procedure provides a simple interface to compile functions and subroutines for use by SAS High-Performance Forecasting. The FCMP procedure accepts a slight variant of the SAS DATA step language. Most features of the SAS programming language can be used in functions and subroutines processed by the FCMP procedure. For more information about the FCMP procedure, see the Base SAS Community at http://support.sas.com/documentation/onlinedoc/base.

For example, the following SAS code creates a user-defined forecasting model for a simple linear trend line called LINEARTREND that is written in the SAS language and stores this subroutine in the catalog SASUSER.HPFUSER. The user-defined forecasting model has the following subroutine signature:

```
SUBROUTINE <SUBROUTINE-NAME> (<ARRAY-NAME>[*], <ARRAY-NAME>[*],
<ARRAY-NAME>[*], <ARRAY-NAME>[*], <ARRAY-NAME>[*]);
```

where the first array, ACTUAL, contains the time series to be forecast, the second array, PREDICT, contains the returned predictions, the third array, STD, contains the returned prediction standard errors, the fourth array, LOWER, contains the returned lower confidence limits, and the fifth array, UPPER, contains the returned upper confidence limits.

```
 proc fcmp outlib=sasuser.hpfuser.funcs;

    subroutine lineartrend( actual[*],
      predict[*], std[*], lower[*], upper[*] );

       nobs = DIM(actual);

n      = 0;
sumx   = 0;
sumx2  = 0;
sumxy  = 0;
sumy   = 0;
sumy2  = 0;
do t = 1 to nobs;
   value = actual[t];
   if nmiss(value) = 0 then do;
             n     = n     + 1;
             sumx  = sumx  + t;
             sumx2 = sumx2 + t*t;
             sumxy = sumxy + t*value;
             sumy  = sumy  + value;
             sumy2 = sumy2 + value*value;
          end;
       end;

det = (n*sumx2 - sumx*sumx);
```

```
constant = (sumx2 * sumy - sumx * sumxy) / det;
slope    = (-sumx  * sumy + n    * sumxy) / det;

  sume2 = 0;
do t = 1 to nobs;
   value = actual[t];
   if nmiss(value) = 0 then do;
   error = value - predict[t];
ume2  = sume2 + error*error;
   end;
end;

  stderr   = sqrt(sume2 / (n-2));
size     = probit(1-0.025/2); /*- 97.5% confidence -*/
width    = size*stderr;

  length = DIM(predict);

do t = 1 to length;
   predict[t] = constant + slope*t;
   std[t]     = stderr;
   lower[t]   = predict[t] - width;
   upper[t]   = predict[t] + width;
end;

    endsub;

 quit;
```

In order to use a user-defined forecasting model, an external forecast model specification must be specified with the same name using the HPFEXMSPEC procedure. For example, the following SAS code creates an external model specification called LINEARTREND and stores this model specification in the catalog SASUSER.MYREPOSITORY. Since the user-defined forecasting model uses two parameters, CONSTANT and SLOPE, the NPARMS=2 option is specified in the EXM statement. The number specified in this option is used for computing statistics of fit (e.g. RMSE, AIC, BIC).

```
proc hpfexmspec modelrepository=sasuser.myrepository
  specname=lineartrend
  speclabel="User defined linear trend";
   exm nparms=2;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. For example, the following SAS code creates a model selection list called MYSELECT and stores this model selection list in the catalog SASUSER.MYREPOSITORY.

```
proc hpfselect modelrepository=sasuser.myrepository
 selectname=myselect
 selectlabel="My Select List";
   spec lineartrend;
run;
```

To use the user-defined forecasting model defined by the FCMP procedure in the HPFENGINE procedure, the CMPLIB option must list the catalogs that contain the SAS language functions and routines. For more information about the FCMP procedure, see the Base SAS Community at http://support.sas.com/documentation/onlinedoc/base.

```
options cmplib = sasuser.hpfuser;
```

At this point,

1. A SAS language subroutine has been defined, LINEARTREND, and stored in the SAS catalog, SASUSER.HPFUSER;
2. An external model specification, LINEARTREND, has been stored in the model repository, SASUSER.MYREPOSITORY;
3. A model selection list, MYSELECT, has been stored in the model repository, SASUSER.MYREPOSITORY;
4. and the search path for the SAS language functions and subroutines has been set to SASUSER.HPFUSER.

The HPFENGINE procedure can now use the user-defined forecasting routine.

For example, the following SAS code forecasts the monthly time series contained in the SASHELP.AIR data set. This data set contains two variables DATE and AIR. The MODELREPOSITORY= SASUSER.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository; the GLOBALSELECTION=MYSELECT options specifies the model selection list; and the USERDEF statement specifies the external/user-defined forecasting model name, LINEARTREND, and variable mapping to the subroutine that must match the subroutine signature. The keyword _PREDICT_ indicates the returned predictions; the keyword _STDERR_ indicates the returned prediction standard errors; the keyword _LOWER_ indicates the returned lower confidence limits; the keyword _UPPER_ indicates the returned upper confidence limits.

```
proc hpfengine data=sashelp.air out=out outfor=outfor outstat=outstat
modelrepository=sasuser.myrepository globalselection=myselect;
   id date interval=month;
   forecast air;
   userdef lineartrend(air, _PREDICT_, _STDERR_, _LOWER_, _UPPER_ );
run;
```

The OUT= data set contains the original data extrapolated by the simple linear trend model (values returned in the _PREDICT_ array) and the OUTFOR= data set contains the forecasts (values returned in the _PREDICT_, _STDERR_, _LOWER_, and _UPPER_ array) and the prediction errors. The OUTSTAT= data set contains the statistics of fit based on the prediction errors and the NPARMS=2 option of the external model specification.

Defining and Using a C Language External Function

The PROTO procedure enables you to register, in batch, external functions written in the C or C++ programming languages for use in SAS. In order to use an external function, it must be called from a SAS language function or subroutine. For more information about the PROTO procedure, see the Base SAS Community at http://support.sas.com/documentation/onlinedoc/base.

For example, the following SAS code creates a user-defined forecasting model for a simple linear trend line called LINEARTREND_C that is written in the C language and stores this external function in the catalog SASUSER.CHPFUSER.

```
    proc proto package=sasuser.chpfuser.cfuncs;

      double lineartrend_c( double * actual,
       int      actualLength,
       double * predict,
       double * std,
       double * lower,
       double * upper,
       int      predictLength );

      externc lineartrend_c;
       double lineartrend_c( double * actual,
         int      actualLength,
         double * predict,
         double * lower,
         double * std,
         double * upper,
         int      predictLength )

   nt       t, n;
   ong      sumx, sumx2;
   ouble    value, sumxy, sumy, sumy2;
   ouble    det, constant, slope, stderr, size, width;
   ouble    error, sume2;

   n      = 0;
   sumx   = 0;
   sumx2  = 0.;
   sumxy  = 0.;
   sumy   = 0.;
   sumy2  = 0.;
```

```
for ( t = 0; t < actualLength; t ++ ) {
    value = actual[t];
    n     = n     + 1;
    sumx  = sumx  + t;
    sumx2 = sumx2 + t*t;
    sumxy = sumxy + t*value;
    sumy  = sumy  + value;
    sumy2 = sumy2 + value*value;
    }

det = (n*sumx2 - sumx*sumx);

constant = (sumx2 * sumy - sumx * sumxy) / det;
slope    = (-sumx  * sumy + n    * sumxy) / det;
  sume2 = 0;
  for ( t = 0; t < actualLength; t ++ ) {
    value = actual[t];
rror = value - predict[t];
    sume2  = sume2 + error*error;
    }

  stderr   = sqrt(sume2 / (n-2.));
size     = 1.96; /*- 97.5% confidence -*/
width    = size*stderr;

for ( t = 0; t < predictLength; t ++ ) {
    predict[t] = constant + slope*t;
    std[t]     = stderr;
    lower[t]   = predict[t] - width;
    upper[t]   = predict[t] + width;
    }
return(0);
}

  externcend;

 run;
```

For example, the following SAS code creates a user-defined forecasting model for a simple linear trend called LINEARTREND and stores this subroutine in the catalog SASUSER.HPFUSER. The catalog SASUSER.CHPFUSER contains functions or subroutines that are used in LINEARTREND. The user-defined forecasting model has the following subroutine signature:

```
SUBROUTINE <SUBROUTINE-NAME> (<ARRAY-NAME>[*], <ARRAY-NAME>[*],
<ARRAY-NAME>[*], <ARRAY-NAME>[*], <ARRAY-NAME>[*]);
```

where the first array, ACTUAL, contains the time series to be forecast, the second array, PREDICT, contains the return predictions, the third array, STD, contains the returned prediction standard errors, the fourth array, LOWER, contains the return lower confidence limits, and the fifth array, UPPER, contains the return upper confidence limits. The LINEARTREND subroutine calls the external function LINEARTREND_C. The DIM function returns the length of the array. For example, the DIM(ACTUAL) function returns the length of the time series array; and the DIM(PREDICT) returns the length of the prediction array. DIM(PREDICT) DIM(ACTUAL) represents the forecast horizon or lead.

```
 proc fcmp outlib=sasuser.hpfuser.funcs inlib=sasuser.chpfuser;

    subroutine lineartrend( actual[*],
predict[*], std[*], lower[*], upper[*] );
ret = lineartrend_c( actual, DIM(actual),
predict, std, lower, upper, DIM(predict));
    endsub;
quit;
```

In order to use a user-defined forecasting model, an external forecast model specification must be specified *with the same name* using the HPFEXMSPEC procedure. For example, the following SAS code creates an external model specification called LINEARTREND and stores this model specification in the catalog SASUSER.MYREPOSITORY. Since the user-defined forecasting model uses two parameters, CONSTANT and SLOPE, the NPARMS=2 option is specified in the EXM statement. The number specified in this option is used in computing statistics of fit.

```
proc hpfexmspec modelrepository=sasuser.myrepository
  specname=lineartrend
  speclabel="User defined linear trend";
   exm nparms=2;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. For example, the following SAS code creates a model selection list called MYSELECT and stores this model selection list in the catalog SASUSER.MYREPOSITORY.

```
proc hpfselect modelrepository=sasuser.myrepository
 selectname=myselect
 selectlabel="My Select List";
   spec lineartrend;
run;
```

To use the user-defined forecasting model defined by the FCMP procedure in the HPFENGINE procedure, the CMPLIB option must list the catalogs that contains the SAS language functions and routines and C language external functions. For more information about the FCMP procedure, see the Base SAS Community at http://support.sas.com/documentation/onlinedoc/base.

```
options cmplib = (sasuser.hpfuser sasuser.chpfuser);
```

At this point,

1. A C language external function has been defined, LINEARTREND_C, and stored in the SAS catalog, SASUSER.CHPFUSER;
2. A SAS language subroutine has been defined, LINEARTREND, which calls the external function, LINEARTREND_C, and stored in the SAS catalog, SASUSER.HPFUSER;
3. An external model specification, LINEARTREND, has been stored in the model repository, SASUSER.MYREPOSITORY;
4. A model selection list, MYSELECT, has been stored in the model repository, SASUSER.MYREPOSITORY;
5. and the search path for the SAS language functions and subroutines has been set to SASUSER.HPFUSER and SASUSER.CHPFUSER.

The HPFENGINE procedure can now use the user-defined forecasting routine.

For example, the following SAS code forecasts the monthly time series contained in the SASHELP.AIR data set. This data set contains two variables DATE and AIR. The MODELREPOSITORY= SASUSER.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository; the GLOBALSELECTION=MYSELECT options specifies the model selection list; and the USERDEF statement specifies the external/user-defined forecasting model name, LINEARTREND, and variable mapping to the subroutine that must match the subroutine signature. The keyword _PREDICT_ indicates the returned predictions; the keyword _STDERR_ indicates the returned prediction standard errors; the keyword _LOWER_ indicates the returned lower confidence limits; the keyword _UPPER_ indicates the returned upper confidence limits.

```
proc hpfengine data=sashelp.air out=out outfor=outfor outstat=outstat
modelrepository=sasuser.myrepository globalselection=myselect;
   id date interval=month;
   forecast air;
   userdef lineartrend(air, _PREDICT_, _STDERR_, _LOWER_, _UPPER_ );
run;
```

The OUT= data set contains the original data extrapolated by the simple linear trend model (values returned in the _PREDICT_ array) and the OUTFOR= data set contains the forecasts (values returned in the _PREDICT_, _STDERR_, _LOWER_, and _UPPER_ array) and the prediction errors. The OUTSTAT= data set contains the statistics of fit based on the prediction errors and the NPARMS=2 option of the external model specification.

Input Time Series Keywords

The user can specify keywords related to the input time series in the USERDEF statement of the HPFENGINE procedure. These keywords specify inputs the user-defined forecasting model. The _TIMEID_ keyword specifies that the time ID values are passed as input arrays to the user-defined forecasting model. The _SEASON_ keyword specifies that the season index values are passed as input arrays to the user-defined forecasting model.

Returned Forecast Component Keywords

At the very least, a user-defined forecasting function or subroutine must return the predictions which is specified by the keyword, _PREDICT_, in the USERDEF statement of the HPFENGINE procedure. The prediction standard errors, lower and upper confidence limits are optional and are specified by the keywords, _STDERR_, _LOWER_, and _UPPER_, respectively. The HPFENGINE procedure will compute the forecasts components that are not returned the user-defined forecasting function based on the external model specification.

Chapter 22
Using External Forecasts

Chapter Contents

Chapter 22
Using External Forecasts

Introduction

External forecasts are forecasts provided by an external source. External forecasts may originate from a statistical model or from another software package, and may have been provided by an outside organization (e.g., marketing organization or government agency), or may be given based solely on judgment.

To use an external forecast in SAS High-Performance Forecasting, an external model specification must be specified using the HPFEXMSPEC procedure. The model specification describes statistical properties of how the forecast was derived. Figure 22.1 describes an example of the ways external forecasts can be specified and used to forecast.

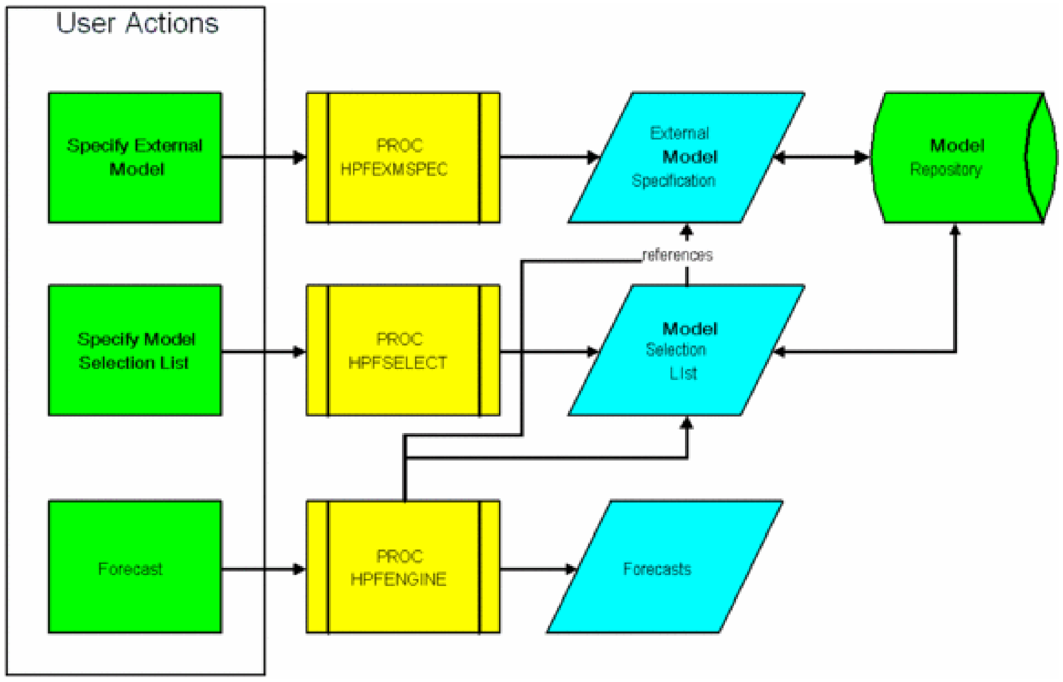


Figure 22.1. Using External Forecasts

Specifying and Using an External Model Specification

The HPFEXMSPEC procedure specifies external models for use by SAS High-Performance Forecasting. The HPFEXMSPEC procedure allows the user to specify information about how the forecasts were derived. This information is used to compute forecast components that are not provided by the user.

For example, the data set SASHELP.AIR contains a monthly time series represented by two variables: DATE and AIR. The following SAS code creates an external forecast in the log transformed metric and stores the external forecasts in the data set WORK.EXTERNALFORECAST. In this example, the HPF procedure is used as the source of the forecasts; but one could imagine that the origin could be any external source.

```
proc timeseries data=sashelp.air out=logair(rename=air=logair);
    id date interval=month;
    var air / transform=log;
run;


proc hpf data=logair out=hpfout outfor=hpfforecast;
    id date interval=month;
    forecast logair / model=addwinters;
run;


data externalforecast; merge sashelp.air hpfforecast;
    drop _NAME_ ACTUAL ERROR;
    by date;
run;
```

The data set WORK.EXTERNALFORECAST contains six variables: DATA, AIR, PREDICT, STD, LOWER and UPPER.

The HPFEXMSPEC procedure can be used to create an external model specification. Continuing the example, the following SAS code creates an external model specification called MYEXTERNAL and stores this model specification in the model repository SASUSER.MYMODELS. The TRANSFORM=LOG option specifies that the external forecasts were generated in the log transformed metric and these forecasts need to be inverse-transformed back to the original metric. The NPARMS=3 option specifies the number of parameters used to generate the external forecasts.

```
proc hpfexmspec modelrepository=sasuser.myrepository specname=myexternal;
    exm transform=log nparms=3;
run;
```

The HPFSELECT procedure can be used to create a model selection list that contains an external model specification as a possible candidate model. Continuing the example, the following SAS code creates a model selection list called MYSELECT and stores this model selection list in the catalog SASUSER.MYREPOSITORY.

```
proc hpfselect modelrepository=sasuser.myrepository
          selectname=myselect
          selectlabel="My Select List";
    spec myexternal;
run;
```

At this point,

1. External forecasts are contained in the data set WORK.EXTERNALFORECAST.
2. An external model specification, MYEXTERNAL, has been stored in the model repository, SASUSER.MYREPOSITORY.
3. A model selection list, MYSELECT, has been stored in the model repository, SASUSER.MYREPOSITORY.

The HPFENGINE procedure can now use both the external model specification and the external forecasts.

Continuing the example, the following SAS code forecasts the monthly time series contained in the WORK.EXTERNALFORECAST data set. The MODELREPOSITORY= SASUSER.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository; the GLOBALSELECTION=MYSELECT options specifies the model selection list; and the EXTERNAL statement specifies the external forecasting model name, LINEARTREND, and variable mapping to the data set variables. The PREDICT= option specifies the variable containing the predictions; the STDERR= option specifies the variable containing the prediction standard errors; the LOWER= option specifies the variable containing the lower confidence limits; the UPPER= option specifies the variable containing the upper confidence limits.

```
 proc hpfengine data=externalforecast
=engout outfor=engforecast outstat=outstat
elrepository=sasuser.myrepository globalselection=myselect;
     id date interval=month;
     forecast air;
     external air=(predict=predict lower=lower upper=upper stderr=std);
 run;
```

The OUT=ENGOUT data set contains the original data extrapolated by the external forecasts and the OUTFOR=ENGFORECAST data set contains the forecasts (values contained in the PREDICT=, STDERR=, LOWER=, and UPPER= data set variables) and the prediction errors. The OUTSTAT=ENGSTAT data set contains the statistics of fit based on the prediction errors and the NPARMS=3 option of the external model specification.

Subject Index

Syntax Index

Your Turn

If you have comments or suggestions about *SAS® High-Performance Forecasting 2.2: User's Guide, Volumes 1 and 2,* 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`**